Adaptive Policies for Scheduling With Reconfiguration Delay: An End-to-End Solution for All-Optical Data Centers

Chang-Heng Wang, *Student Member, IEEE*, and Tara Javidi, *Senior Member, IEEE*

***Abstract*— All-optical switching networks have been considered a promising candidate for the next generation data center networks thanks to its scalability in data bandwidth and power efficiency. However, the bufferless nature and the nonzero reconfiguration delay of optical switches remain great challenges in deploying all-optical networks. This paper considers the end-to-end scheduling for all-optical data center networks with no in-network buffer and nonzero reconfiguration delay. A framework is proposed to deal with the nonzero reconfiguration delay. The proposed approach constructs an adaptive variant of any given scheduling policy. It is shown that if a scheduling policy guarantees its schedules to have schedule weights close to the MaxWeight schedule (and thus is throughput optimal in the zero reconfiguration regime), then the throughput optimality is inherited by its adaptive variant (in any nonzero reconfiguration delay regime). As a corollary, a class of adaptive variants of the well-known MaxWeight policy is shown to achieve throughput optimality without prior knowledge of the traffic load. Furthermore, through numerical simulations, the simplest such policy, namely, the Adaptive MaxWeight, is shown to exhibit better delay performance than all prior work.**

***Index Terms*— Reconfiguration delay, scheduling, throughput optimality, drift analysis.**

I. Introduction

MASSIVE data centers serve as the basis of a huge variety of online services and applications nowadays. The underlying network interconnects face increasingly stringent performance requirements such as high data bandwidth and low latency. All-optical networks emerge as a promising candidate for the next generation data center networks and benefit from two technical breakthroughs: (1) the advancement of dense wavelength division multiplexing (DWDM) in optical fibers and (2) the optical switches substituting traditional electronic switches, which typically incur high cost and high power demand when supporting high data bandwidth. However, due to the inherently bufferless nature of optical switches, the data transmission in an all-optical network will need to be conducted in an end-to-end fashion. In other words, all of the buffering occurs at the end hosts or the top of the rack (ToR) switches, and the data transmission would take place only when the connection between hosts is established. An example of this architecture is proposed in [1], in form of a prototype design with extremely promising properties. From an end-to-end perspective, this architecture with zero in-network buffering can be viewed as a single crossbar switch interconnecting the end hosts (or ToR switches), with the exception that the full bisection bandwidth is not always guaranteed. Under this architecture, an efficient utilization of the all-optical network depends on a centralized controller that can efficiently schedule the end-to-end transmissions.

The main challenge of the scheduling for optical networks, as opposed to the traditional electronic crossbar switch scheduling, comes from the fact that optical networks typically exhibit a nonzero *reconfiguration delay* upon changing the circuit configuration. For instance, candidate technologies such as MEMS [2], WSS [3] involve mechanically directing laser beams and thus require a certain time to finish circuit reconfiguration due to physical limits. During the circuit reconfiguration, reliable packet transmission could not be supported in the network. Notice that for the architecture considered in this paper, the reconfiguration delay may further include the time for the control plane to control/communicate with the optical switches, and the time for end hosts to initiate/pause packet transmissions. For example, a state of the art system implemented in [1] reports a measured reconfiguration delay up to 20 μs (including delay caused by the control plane and end-host), which is significantly larger than the inter-frame gap of 0.96 ns (for 100 Gigabit Ethernet). This nonzero reconfiguration delay motivates the need for scheduling policies that explicitly account for the reconfiguration delay.

With the presence of the reconfiguration delay, it is well known that the scheduling policy should avoid reconfiguring the circuit too frequently. In fact, most of the prior work for scheduling policies with nonzero reconfiguration delay ([4]–[6]) tend to provision the future schedules for a certain time period (usually a long time period) in order to control the frequency of reconfiguration. These scheduling policies, classified as "quasi-static" policies, however, may perform poorly since their schedules may depend on severely out-dated queue length information. In contrast, "dynamic" policies determine each schedule based on the most up-to-date queue length information. An example subclass of dynamic policies is the frame-based policies such as the Fixed-Frame MaxWeight (FFMW) [7], which has good delay performance under nonzero reconfiguration delay, assuming the arrival

Manuscript received November 11, 2015; revised May 16, 2016 and October 16, 2016; accepted November 23, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Mellia. This work was supported by L3 Communications, NSF Center for Integrated Access Networks through the Event-Based Information Acquisition, Learning, and Control in High-Dimensional Cyber-Physical Systems under NSF Grant EEC-0812072 and Grant CNS-1329819.

The authors are with the University of California, San Diego, CA 92093 USA (e-mail: chw009@ucsd.edu; tjavidi@ucsd.edu).

Digital Object Identifier 10.1109/TNET.2016.2644617

statistics is known in advance. These frame-based policies require the duration of the frame to be set to ensure that the reduced duty cycle due to the reconfiguration delay could sustain the traffic load. As a result, these policies require prior knowledge of the arrival statistics to ensure the stability of the network, which may be impractical in many cases.

On the other hand, it has been shown that scheduling policies may achieve optimal throughput in the nonzero reconfiguration regime while requiring no prior knowledge of the arrival statistics, *e.g.* the Adaptive MaxWeight (AMW) policy [8] and the Switching Curve Based (SCB) policy [9]. These policies exhibit the behavior of adapting the frequency of reconfiguration with respect to the queue lengths, which has also been shown in the Adaptive Back-pressure [10] proposed in the context of single-commodity, multi-hop networks. In this paper, we extend the idea of this adaptive mechanism and propose a framework that could transform a broad variety of scheduling policies that are throughput optimal under zero reconfiguration delay into throughput optimal policies under nonzero reconfiguration delay, which are referred as adaptive policies. The adaptive policies make scheduling decisions every time slot and determines both the schedule and the time to reconfigure the schedule based on the most recent queue length information. The main idea behind the adaptive policy is to keep the current schedule as long as it is "good enough" so that the schedule reconfiguration only occurs when it is necessary. To be more specific, the construction of an adaptive policy requires a scheduling policy π, a weight function W to evaluate schedules, and a sublinear function g. At each time slot t, the policy π proposes a schedule $\mathbf{\Pi}(t)$, and the adaptive policy computes the schedule weight difference between $\mathbf{\Pi}(t)$ and the current schedule. If the weight difference is less than a threshold dependent on the function g, then the current schedule is considered good enough; otherwise the schedule is reconfigured to $\mathbf{\Pi}(t)$. The constructed policy is called the g-adaptive variant of π where the function g measures the reluctance of changing the schedule, and is thus referred to as the *hysteresis function*. We show in this paper that if the original policy π has schedule weight that is close to the MaxWeight schedule (either in deterministic or expected sense), which guarantees throughput optimality under a zero reconfiguration delay, then its g-adaptive variant achieves throughput optimality under any fixed reconfiguration delay. Note that this stability guarantee does not require prior knowledge on the arrival statistics or the value of reconfiguration delay to stabilize the network, as opposed to the frame-based policies, such as the FFMW policy.

While we present our work in the context of all-optical networks, the end-to-end scheduling model and the proposed adaptive policies may also be applicable in other contexts. For example, in the hybrid (optical-electrical) network topologies such as that proposed in [11] and [12], the adaptive policies may complement the scheduling of the optical circuit part. As another example, note that the notion of end host buffering and end-to-end scheduling has also been explored in the regime of electronic packet-switched data center network [13] recently. This suggests that our proposed scheduling policies can also be utilized in the context of electronic packet switches in order to further reduce delay and improve performance.

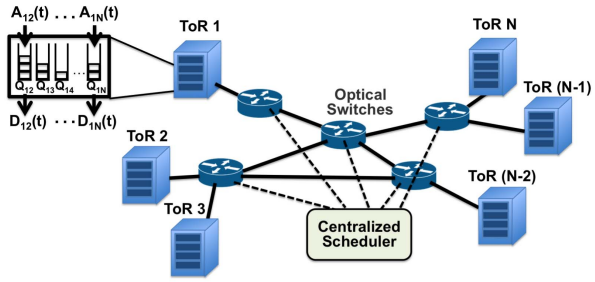

Fig. 1. An illustration of the system model.

The rest of the paper is organized as follows. In the next section, the network model and the notion of stability are introduced. In section III, we introduce the class of adaptive policies and analyze its throughput as well as certain delay bounds. We then explain the mechanism of the adaptive policies and compare them with scheduling policies in the literature qualitatively in section IV. Section V gives the performance evaluation and the comparison between scheduling policies through simulations. Finally, we conclude with a summary and some future directions in section VI.

II. System Model

A. The Network Model

Consider a set of N top of rack (ToR) switches, labeled by $\{1, 2, \ldots, N\}$, which are interconnected by an optical switched network, as shown in Fig. 1. Each ToR switch can serve both as a source and a destination simultaneously. We assume no buffering in the optical network, hence all the buffering occurs in the edge of the network, *i.e.* within the ToR switches. Each ToR switch maintains $N-1$ edge queues (either physically or virtually), which are denoted by Q_{ij}, where $j \in \{1, 2, \ldots, N\} \backslash \{i\}$. Packets going from ToR switch i to j are enqueued in the edge queue Q_{ij} before transmission.

The system considered is assumed to be time-slotted, with the time indexed as $t \in \mathbb{N} = \{0, 1, 2, \ldots\}$. Each slot duration is the transmission time of a single packet, which is assumed to be a fixed value. Let $A_{ij}(t)$ and $D_{ij}(t)$ be the number of packets arrived at and departed from queue Q_{ij} at time t, respectively. Let $L_{ij}(t)$ be the number of packets in the edge queue Q_{ij} at the beginning of the time slot t. For ease of notation, we write $\mathbf{A}(t) = [A_{ij}(t)], \mathbf{D}(t) = [D_{ij}(t)], \mathbf{L}(t) = [L_{ij}(t)]$, where $\mathbf{A}(t), \mathbf{D}(t), \mathbf{L}(t) \in \mathbb{N}^{N \times N}$. We adopt the convention that the packet arrivals occur at the end of each time slot, and the queue dynamics is then given by

$$L_{ij}(t+1) = L_{ij}(t) - D_{ij}(t) + A_{ij}(t)$$

We assume the arrival processes $A_{ij}(t)$ to be independent over $i, j \in \{1, 2, \ldots, N\}, i \neq j$. Each process $A_{ij}(t)$ is i.i.d. over time slots. Let the mean of $A_{ij}(t)$ be the traffic rate $\lambda_{ij} = \mathbb{E}[A_{ij}(0)]$, and define the traffic rate matrix $\boldsymbol{\lambda} = [\lambda_{ij}] \in \mathbb{R}_+^{N \times N}$.

B. Schedules and Scheduling Policies

Let $\mathbf{S}(t) \in \{0, 1\}^{N \times N}$ denote the schedule at time t, which indicates the optical circuits established between the

ToR switches. We set $S_{ij}(t) = 1$ if an optical circuit from ToR i to ToR j exists at time t, and $S_{ij}(t) = 0$ otherwise. The feasible schedules for the network are determined by the network topology and physical constraints on simultaneous data transmissions. We let $\mathcal{S}$ denote the set of all feasible schedules, *i.e.* $\mathbf{S}(t) \in \mathcal{S}$ for all t. For instance, it's common to assume at any t each ToR can only transmit to at most one destination, and can only receive from at most one source, i.e. $\sum_i S_{ij}(t) \leq 1, \sum_j S_{ij}(t) \leq 1$. Under such assumption $\mathcal{S} \subset \mathcal{P}$, where $\mathcal{P}$ is the set of all permutation matrices. Note the permutation matrices might not all be feasible in a network; however, if all such schedules are in the feasible schedule set $\mathcal{S}$, we say the network topology is non-blocking.

Upon reconfiguring a schedule, the network incurs a reconfiguration delay, during which no packet could be transmitted. In this paper we focus on the effect of the reconfiguration delay on the performance of scheduling policies. We make this notion formal through the following two definitions:

Definition 1: Let $\{t_k^S\}_{k=1}^{\infty}$ denote the time instances when the schedule is reconfigured. The schedule between two schedule reconfiguration time instances remains the same, *i.e.*

$$\mathbf{S}(\tau) = \mathbf{S}(t_k^S), \quad \forall \tau \in [t_k^S, t_{k+1}^S - 1]$$

Definition 2: Let Δ_r be the reconfiguration delay associated with reconfiguring the schedule of the network. During the period of schedule reconfiguration, no packet transmission could occur in the network. This means that $\forall i, j \in \{1, 2, \dots N\}$, $\forall k \in \mathbb{N}_+$, and $0 \leq \tau \leq \Delta_r$, we have $D_{ij}(t_k^S + \tau) = 0$. Note that for all other time, $D_{ij}(t) = S_{ij}(t)$ if $L_{ij}(t) > 0$.

The schedule of the network is determined by a *scheduling policy*. In this paper, we restrict our discussions only to Markov scheduling policies defined as below.

Definition 3: A **Markov scheduling policy** determines the schedules $\{\mathbf{S}(t)\}_{t=0}^{\infty}$, and at each time t, the schedule $\mathbf{S}(t) \in \mathcal{S}$ depends on the history solely through the current state $X_t = \Big(\mathbf{S}(t-1), \mathbf{L}(t)\Big)$. Restricting attention to this class of policies ensures the process $\{X_t\}_{t=0}^{\infty}$ to be a Markov process. In this case we denote the Markov scheduling policy as π and use the notation $\mathbf{\Pi}(t)$ to denote the choice of schedule generated by π given the state $X_t = \Big(\mathbf{S}(t-1), \mathbf{L}(t)\Big)$.

Notice that we intentionally separate the notation to let $\mathbf{S}(t)$ denote the schedule of the network and $\mathbf{\Pi}(t)$ denote the schedule generated by the Markov policy π.

A specific scheduling policy of interest to our work is the MaxWeight policy defined below.

Definition 4: Given a schedule $\mathbf{S} \in \mathcal{S}$, the **weight** of the schedule $\mathbf{S}$ is a continuous, non-negative real function on the queue lengths, $W_{\mathbf{S}} : \mathbb{N}^{N \times N} \to \mathbb{R}_+$ with $W_{\mathbf{S}}(\mathbf{0}) = 0$. The **weight function** is defined as $W : \mathcal{S} \times \mathbb{N}^{N \times N} \to \mathbb{R}$ with $W(\mathbf{S}, \cdot) = W_{\mathbf{S}}(\cdot)$.

We also abuse the notation to let $W_{\mathbf{S}}(t) = W_{\mathbf{S}}(\mathbf{L}(t))$ denote the weight of schedule S at time t whenever there is no confusion.

Definition 5: Given a weight function W, the **MaxWeight** policy determines the schedule $\mathbf{\Pi}^*(t)$ as the schedule that has the maximum weight among feasible schedules at time t, *i.e.*

$$\mathbf{\Pi}^*(t) = \arg\max_{\mathbf{S} \in \mathcal{S}} W_{\mathbf{S}}(t).$$

We also define the **maximum weight** at time t as the weight of the MaxWeight schedule at time t, *i.e.* $W^*(t) = \max_{\mathbf{S} \in \mathcal{S}} W_{\mathbf{S}}(t)$.

C. Network Stability

Definition 6: The network is **strongly stable** under policy π or a policy π is said to strongly stabilize the network if its queue lengths $\mathbf{L}(t)$ satisfies:

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\big[||\mathbf{L}(\tau)||\big] < \infty$$

where $||\mathbf{L}(t)|| = \sum_{i,j=1}^{N} L_{ij}(t)$ is the total queue length of the system and the expectation is taken with respect to the statistics induced by random packet arrivals and policy π. This means that a scheduling policy directly controls the statistics of queue occupancies and hence the per packet delay.

With the notion of the strong stability, we may then define the admissible arrival traffic and the throughput optimality of a scheduling policy as follows:

Definition 7: The arrival traffic $\mathbf{A}(t)$ is **admissible** if there exists a scheduling policy which strongly stabilizes the system. The traffic rate matrix $\boldsymbol{\lambda}$ is admissible if $\mathbf{A}(t)$ is admissible.

Definition 8: The **capacity region** is the set of all admissible traffic rate matrices and is denoted as $\mathbf{\Lambda}$.

We may then define the traffic load as follows:

Definition 9: The **load** of traffic rate matrix $\boldsymbol{\lambda}$ is defined as

$$\rho(\boldsymbol{\lambda}) = \inf\{r : \boldsymbol{\lambda} \in r\bar{\mathbf{\Lambda}},\ 0 < r < 1\}$$

where $\bar{\mathbf{\Lambda}}$ is the closure of $\mathbf{\Lambda}$. We shall use ρ instead of $\rho(\boldsymbol{\lambda})$ whenever there is no confusion.

Definition 10: A scheduling policy achieves **throughput optimality** if it strongly stabilizes the network under any admissible arrival traffic.

When $\Delta_r = 0$, several scheduling policies (e.g. [14]–[16]) have been shown to achieve throughput optimality in the literature. However, in the regime of $\Delta_r > 0$, these scheduling policies typically lose the throughput optimality guarantee since they do not address the fact that each schedule reconfiguration would result in a significant reduction in the duty cycle and hence decrease the utility of the network. The challenge for scheduling in the $\Delta_r > 0$ regime is that both the quality of the schedules and the rate of schedule reconfiguration affects the performance of a scheduling policy. It is not hard to see that there is a tradeoff between these two factors: Reducing the rate of reconfiguration means that the network is forced to stick with a schedule longer and loses the chance to use a better schedule; on the other hand, pursuing a better schedule most of the time inevitably increases the rate of reconfiguration and thus the incurred overhead. Therefore, a good scheduling policy in the $\Delta_r > 0$ regime must strive to achive the balance between these two factors.

III. Main Results

In this section, we introduce a broad class of adaptive scheduling policies that balance the rate of schedule reconfiguration and the quality of the schedules without prior knowledge

of the arrival traffic. We first introduce a general approach that could transform any scheduling policy to an adaptive variant of the original policy and give some example policies generated by this approach. We then introduce several adaptive policies that could achieve throughput optimality in the $\Delta_r > 0$ regime. In particular, we primarily focus on scheduling policies that have the schedule weight close to the MaxWeight policy (either in deterministic sense or in expectation). These scheduling policies have been shown in the literature (e.g. [17]) to achieve throughput optimality when $\Delta_r = 0$. In this paper, we show that under mild conditions, the adaptive variants of these policies achieve throughput optimality under any fixed reconfiguration delay $\Delta_r > 0$.

A. Adaptive Policies

Given a Markov scheduling policy π and current state X_t, let $\mathbf{\Pi}(t) = \pi(X_t)$ denote the schedule generated by π at time t. With the weight function W, let $W^\pi(t) = W_{\mathbf{\Pi}(t)}(t)$ denote the weight of the schedule $\mathbf{\Pi}(t)$ at time t.

Now at any given time t, $\Delta W(t) = W^\pi(t) - W_{\mathbf{S}(t-1)}(t)$ measures the potential improvement (in terms of schedule weight) associated with following policy π instead of sticking with the previous schedule $\mathbf{S}(t-1)$. Since each schedule change results in a loss in duty cycle, the proposed class of adaptive policies show some inertia against frequent schedule reconfiguration. More precisely, let us define a **hysteresis function** g where $g : \mathbb{R} \to \mathbb{R}$ is a nonnegative, continuous, strictly increasing, and sublinear (*i.e.* $\lim_{x\to\infty} \frac{g(x)}{x} = 0$) function. Our proposed g-adaptive Markov policy π^g uses the new schedule $\mathbf{\Pi}(t)$ only if $\Delta W(t) > g(W^\pi(t))$. In other words,

$$\mathbf{\Pi}^g(t) = \pi^g\Big(\mathbf{S}(t-1), \mathbf{L}(t)\Big) = \begin{cases} \mathbf{S}(t-1) & \text{if } \Delta W(t) \le g(W^\pi(t)) \\ \mathbf{\Pi}(t) & \text{if } \Delta W(t) > g(W^\pi(t)) \end{cases}$$

Note that the proposed g-adaptive Markov policy could be constructed from any Markov scheduling policy. Given the Markov policy π, we call π^g the g-adaptive variant of π.

The intuition behind this construction is that the g-adaptive policy holds on to the previous schedule as long as it is "good enough" relative to the current schedule generated by π, $\Pi(t)$. The sublinearity of the function $g(\cdot)$ is a technical assumption in order to achieve the throughput optimality, which would become clear in the analysis given in the next subsection.

We now give some examples for possible combinations of hysteresis function $g(\cdot)$ and scheduling policy π. We leave the analysis of throughput optimality to the next subsection.

Example 1 (Adaptive MaxWeight Policy [8]): Let $W_{\mathbf{S}}(\mathbf{L}) = \sum_{i,j=1}^N S_{ij}L_{ij}$, and the scheduling policy π be the MaxWeight policy. Suppose the function $g(\cdot)$ takes the form $g(x) = (1-\gamma)x^{1-\delta}$, where $\gamma \in (0,1), \delta \in (0,1)$. Then the g-adaptive variant of π is called the Adaptive MaxWeight, as introduced in [8].

The Adaptive MaxWeight policy computes the weight difference $\Delta W = W^*(t) - W_{\mathbf{S}(t-1)}(t)$ to the threshold $g(W^*(t)) = (1-\gamma)(W^*(t))^{1-\delta}$ at each time slot. It reconfigures to the MaxWeight schedule if the difference is above the threshold, otherwise keeps the current schedule.

Remark 1 Notice that the Switching Curve Based (SCB) policy proposed in [9] is similar to the Adaptive MaxWeight policy, except that in SCB policy, the threshold for reconfiguration is taken as a sublinear function of the total queue length, instead of the maximum weight. Due to the relation between the maximum weight and the total queue length ($\frac{1}{N}||\mathbf{L}(t)|| \le W^*(t) \le ||\mathbf{L}(t)||$), the two policies behave similarly. We only introduce the Adaptive MaxWeight here because it fits better in the framework of adaptive policies introduced here.

Example 2 (g-Adaptive Variant of the Pipelined Maxweight Policy): Under the pipelined MaxWeight policy [18], the scheduler determines the schedule at time t to be the schedule maximizing the weight at time $t-K$, for some fixed scalar $K < \infty$. Intuitively, one can think of the pipelined MaxWeight as a scheduler that initiates MaxWeight computation at each time slot but obtains and enforces it only K time slots later. Therefore, the schedule at time t is the MaxWeight schedule based on $\mathbf{L}(t-K)$. The selection of the function $g(\cdot)$ could be any continuous, strictly increasing, and sublinear function, e.g. $g(x) = \log(1+x)$.

In the following three examples, we introduce scheduling policies that are MaxWeight policies with respect to different definitions of the weight function W. For ease of presentation, we consider the hysteresis function to be $g(x) = (1-\gamma)x^{1-\delta}$ as in Example 1.

Example 3 (g-Adaptive Variant of the Projective Cone Scheduling Policy): In [19], the weight function W takes the form $W_{\mathbf{S}}(\mathbf{L}) = \Big\langle vec(\mathbf{S}), \mathbf{P}vec(\mathbf{L})\Big\rangle$ where $P \in \mathbb{R}^{N^2\times N^2}$ and $vec(\cdot)$ is the vectorization of a matrix. The MaxWeight policy corresponding to this weight function W is called the Projective Cone Scheduling (PCS) policy. It was shown that if $\mathbf{P}$ is positive definite, symmetric, and has non-positive off-diagonal elements, then the PCS policy is throughput optimal.

Example 4 (g-Adaptive Variant of the General Prioritized Scheduling Policy): In [20], a general class of scheduling policies based on queue prioritization and queue grouping is proposed under the setting of multi-commodity multi-hop network (which includes the crossbar switch as a special case). The queue prioritization results in the partition of $\mathbb{R}^{N^2}$ into a finite number of cones, and the weight function of each schedule $\mathbf{S} \in \mathcal{S}$, $W_{\mathbf{S}}$, is then a linear function on each of these cones, and a continuous, piecewise linear function on $\mathbb{R}^{N^2}$.

Example 5 (g-Adaptive Variant of the MaxWeight-α Policy): For any fixed $\alpha > 0$, let the weight function be characterized by $W_{\mathbf{S}}(\mathbf{L}) = \sum_{i,j=1}^N L_{ij}^\alpha(t)S_{ij}(t)$, then the corresponding MaxWeight policy is referred as MaxWeight-α policy in the literature. Under the MaxWeight-α policy, the schedule at time t is given by

$$\mathbf{\Pi}_{MW\alpha}(t) = \arg\max_{\mathbf{S}\in\mathcal{S}} \sum_{i,j=1}^N L_{ij}^\alpha(t)S_{ij}(t)$$

Since the MaxWeight scheduling policy is well known for its high computation complexity, in the literature several

lower complexity policies have also been proposed with good stability conditions when $\Delta_r = 0$. We consider the adaptive variant of these policies as well.

Example 6 (g-Adaptive Variant of Tassiulas Random Policy): The Tassiulas Random policy [15] utilizes random schedule selection and memory to determine the schedule. It compares the weight between the last schedule and a randomly selected schedule (according to an arbitrary distribution on the feasible schedule $\mathcal{S}$, say uniformly random). Let $\mathbf{Z}(t)$ be the randomly selected schedule at time t, then the schedule determined by the Tassiulas random policy is given by

$$\mathbf{\Pi}_Z(t) = \begin{cases} \mathbf{\Pi}_Z(t-1) & \text{if } W_{\mathbf{\Pi}_Z(t-1)}(t) \geq W_{\mathbf{Z}(t)}(t) \\ \mathbf{Z}(t) & \text{otherwise} \end{cases}$$

Example 7 (g-Adaptive Variant of the Hamiltonian Policy): The Hamiltonian policy (ALGO 3 in [16]) utilizes the Hamiltonian walk on the set of permutation matrices and memory to determine the schedule. It compares the schedule weight between the last schedule and the schedule on the Hamiltonian path and select the schedule with higher weight. Specifically, let $\mathbf{H}(t)$ be the schedule on the Hamiltonian path at time t, then the schedule determined by the Hamiltonian policy is given by

$$\mathbf{\Pi}_H(t) = \begin{cases} \mathbf{\Pi}_H(t-1) & \text{if } W_{\mathbf{\Pi}_H(t-1)}(t) \geq W_{\mathbf{H}(t)}(t) \\ \mathbf{H}(t) & \text{otherwise} \end{cases}$$

B. Drift Analysis: Lipshitz Continuous Weight Function

Before we begin the analysis of the adaptive policies, we state the following assumption on the arrival traffic:

Assumption 1: Assume the arrival at each queue is bounded from above, i.e. $\mathbf{A}_{ij}(t) \leq A_{max} < \infty, \forall i, j \in \{1, 2, \ldots, N\}, \forall t$.

For the weight function, we first focus only on Lipshitz continuous weight functions, and consider more general cases in later discussions:

Assumption 2: Assume the weight function $W : \mathcal{S} \times \mathbb{R}^{N^2} \to \mathbb{R}$ *satisfies that, for each schedule* $\mathbf{S} \in \mathcal{S}$*, the weight of the schedule,* $W_{\mathbf{S}} : \mathbb{R}^{N^2} \to \mathbb{R}$*, is Lipschitz continuous.* [1]

We utilize the Foster-Lyapunov Theorem (cf. Fact 1 in Appendix) to show the throughput optimality of the adaptive policies. Let $V : \mathbb{R}^{N \times N} \to \mathbb{R}$ be a non-negative, real-valued Lyapunov function, and define the drift of V at time t as $\Delta V(t) = V(\mathbf{L}(t+1)) - V(\mathbf{L}(t))$. The general procedure of establishing the throughput optimality is to show that the conditional Lyapunov drift $\mathbb{E}\left[\Delta V(t) | \mathbf{L}(t)\right]$ to be negative in all but a finite subset of the states $\mathbf{L}(t)$. The following assumption characterizes the Lyapunov functions (with respect to a weight function) used for throughput analysis of most of the scheduling policies under zero reconfiguration delay.

Assumption 3: Given the weight function W*, the non-negative, real-valued Lyapunov function* $V : \mathbb{R}^{N \times N} \to \mathbb{R}$ *satisfies that, under the schedule* $\mathbf{S}$*, the expected Lyapunov drift satisfies:*

$$\mathbb{E}\left[\Delta V(t) \middle| \mathbf{L}(t), \mathbf{S}(t) = \mathbf{S}\right] \leq \Lambda(\mathbf{L}(t)) - W_{\mathbf{S}}(\mathbf{L}(t)) \quad (1)$$

where $\Lambda(\cdot)$ *is a term that is not dependent on* $\mathbf{S}$*.*

Furthermore, suppose the schedule $\mathbf{S}$ *is the MaxWeight schedule* $\mathbf{\Pi}^*(t)$*, then*

$$\Lambda(\mathbf{L}(t)) - W^*(t) \leq -\epsilon W^*(t) + K \quad (2)$$

for some fixed constant $\epsilon, K > 0$*.*

Note that (1) determines an upper bound on the conditional Lyapunov drift which depends on the schedule $\mathbf{S}$ only through the weight function $W_{\mathbf{S}}$, and (2) is needed to establish the negative drift. We use the MaxWeight policy as an example to illustrate Assumption 3. Take the weight function as $W_{\mathbf{S}}(\mathbf{L}) = \langle \mathbf{S}, \mathbf{L} \rangle$ and the Lyapunov function as $V(\mathbf{L}) = \langle \mathbf{L}, \mathbf{L} \rangle$. We have the conditional Lyapunov drift given by

$$\mathbb{E}\left[\Delta V(t) \middle| \mathbf{L}(t)\right] \leq \langle \boldsymbol{\lambda}, \mathbf{L}(t) \rangle - W_{\mathbf{S}}(t) + N^2 A_{\max}^2,$$

hence we have $\Lambda(\mathbf{L}(t)) = \langle \boldsymbol{\lambda}, \mathbf{L}(t) \rangle + N^2 A_{\max}^2$. Also, with $\mathbf{S} = \mathbf{\Pi}^*(t)$, we have $\langle \boldsymbol{\lambda}, \mathbf{L} \rangle - W^*(t) \leq -(1-\rho) W^*(t)$. Hence

$$\Lambda(\mathbf{L}(t)) - W^*(t) \leq -(1-\rho) W^*(t) + N^2 A_{\max}^2$$

Remark: Note that the conditional Lyapunov drift expression in Assumption 3 is evaluated under zero reconfiguration delay. In the case of nonzero reconfiguration delay and the network is in reconfiguration at time t, we may simply evaluate the conditional drift in (1) with $W_{\mathbf{S}}(\mathbf{L}(t)) = 0$, indicating the schedule is not serving the network.

With the previous assumptions, we now introduce a class of scheduling policies that guarantee bounded expected weight differences to the MaxWeight schedule at all times, and show the throughput optimality of their g-adaptive variant.

Condition 1: Given the weight function W*, the scheduling policy* π *satisfies the following property:*

There exists a constant $G < \infty$ *such that the schedule weight of the scheduling policy* π*,* $W^\pi(t) = W_{\mathbf{\Pi}(t)}(t)$*, satisfies*

$$\mathbb{E}^\pi\left[W^\pi(t) | \mathbf{L}(t)\right] \geq W^*(t) - G, \quad \forall t$$

Theorem 1: Given any reconfiguration delay $\Delta_r > 0$*. Assume the arrival traffic is admissible and satisfies Assumption 1. Assume the weight function* W *satisfies Assumption 2 and there exists a non-negative, real-valued Lyapunov function* $V : \mathbb{R}^{N \times N} \to \mathbb{R}$ *that satisfies Assumption 3 with weight function* W*.*

Suppose the Markov policy π *satisfies Condition 1 with weight function* W*, then the g-adaptive variant of* π *is throughput optimal.*

The proof of Theorem 1 is given in Appendix A. The proof utilizes the Foster-Lyapunov Theorem and consists of two main components. The first one shows that the schedule of the adaptive policy π^g has weight difference to the MaxWeight schedule bounded by a sublinear function of the maximum weight. This potentially gives the negative Lyapunov drift, as similarly argued in [18]. The second part is that the rate of schedule reconfiguration becomes smaller as the queue lengths

[1] Given X, Y metric spaces with corresponding metric d_X, d_Y, a function $F : X \to Y$ is Lipschitz continuous if there exists a real constant $B \geq 0$ such that $d_Y(F(x_1), F(x_2)) \leq B d_X(x_1, x_2)$.

become larger. With this property, we show that the overhead incurred by the schedule reconfiguration delay becomes arbitrarily small when the total queue lengths $||\mathbf{L}(t)||$ increases. This suffices to give the guarantee of negative expected drift when $||\mathbf{L}(t)||$ is large and thus guarantee the stability.

With Theorem 1, we are now ready to show the throughput optimality of some example adaptive policies given in the previous subsection.

Corollary 1: The adaptive policies in examples 1 and 2 achieve throughput optimality.

Proof: With the example weight function given for Assumption 2 and 3, we have these assumptions satisfied. Now for example 1, since π is the MaxWeight policy, Condition 1 is satisfied by definition, with $G = 0$. As for example 2, Condition 1 is satisfied with $G = N(A_{\max} + 1)K$. To see this, recall that $W^{\pi}(t) = W_{\mathbf{\Pi}(t)}(\mathbf{L}(t)) = \left\langle \mathbf{L}(t), \mathbf{\Pi}(t) \right\rangle$, and note that at most one packet could depart from each queue at each time slot, and hence

$$\begin{aligned} \left\langle \mathbf{L}(t), \mathbf{\Pi}(t) \right\rangle &\geq \left\langle \mathbf{L}(t-K), \mathbf{\Pi}(t) \right\rangle - NK \\ &= \left\langle \mathbf{L}(t-K), \mathbf{\Pi}^*(t-K) \right\rangle - NK. \quad (3) \end{aligned}$$

Since the arrivals are bounded by $A_{\max}$, we also have

$$\left\langle \mathbf{L}(t), \mathbf{\Pi}^*(t) \right\rangle \leq \left\langle \mathbf{L}(t-K), \mathbf{\Pi}^*(t-K) \right\rangle + NA_{\max}K. \quad (4)$$

From (3) and (4), $W^{\pi}(t) \geq W^*(t) - N(A_{\max} + 1)K$ and thus Condition 1 is satisfied. ∎

Corollary 2: The adaptive policies in examples 3 and 4 achieve throughput optimality.

Proof: We first check that the weight functions satisfy Assumption 2. For the PCS policy, given $W_{\mathbf{S}}(\mathbf{L}) = \langle vec(\mathbf{S}), \mathbf{P}vec(\mathbf{L}) \rangle$, let $P_{\max} = \max_{i,j} |P_{ij}|$, then we have $W_{\mathbf{S}}(\mathbf{L}_1) - W_{\mathbf{S}}(\mathbf{L}_2) = \langle vec(\mathbf{S}), \mathbf{P}\left(vec(\mathbf{L_1}) - vec(\mathbf{L_2})\right) \rangle \leq N^2 P_{\max} ||\mathbf{L}_1 - \mathbf{L}_2||$, for $\mathbf{L}_1, \mathbf{L}_2 \in \mathbb{R}^{N \times N}$. For the general prioritized policy, $W_{\mathbf{S}}(\cdot)$ is linear in each partitioned cone of $\mathbb{R}^{N \times N}$, hence is Lipshitz continuous. Since the number of partitioned cones is finite, taking the B as the maximum of the Lipshitz constants over all the cones, then $W_{\mathbf{S}}(\cdot)$ is Lipshitz continuous with Lipshitz constant B.

Since the PCS policy and the general prioritized policy are MaxWeight policies with respect to their corresponding weight function, Condition 1 is trivially satisfied. Also, with their corresponding Lyapunov functions in [19] and [20], Assumption 3 is also satisfied. We then have the throughput optimality. ∎

Corollary 3: The g-adaptive variant of the Hamiltonian policy given in example 1 and the g-adaptive variant of the Tassiulas random policy given in example 6 are throughput optimal.

Proof: In [16] the Hamiltonian policy is shown to satisfy Condition 1 with $G = 2N(N!)$, hence by Theorem 1 it achieves throughput optimality.

For the Adaptive Tassiulas, let $\Pr\{\mathbf{Z}(t) = \mathbf{\Pi}^*(t)\} \geq \epsilon > 0$ for all t and for some ϵ. In [21] the Tassiulas random policy is shown to satisfy Condition 1 with $G = 2N\frac{1-\epsilon}{\epsilon}$, hence by Theorem 1 it achieves throughput optimality. ∎

C. Drift Analysis: Non-Lipshitz Continuous Weight Function

In this subsection, we extend the throughput analysis for some non-Lipshitz continuous weight functions. In particular, we consider weight functions that take the following form:

Assumption (2'): Assume the weight function W is defined as

$$W_{\mathbf{S}}(\mathbf{L}) = \sum_{i,j=1}^{N} f(L_{ij}) S_{ij}$$

for some continuous, non-decreasing function f with $f(0) = 0$.

For the ease of exposition, we restrict the policies of interest to those that have strictly bounded weight difference to the MaxWeight schedule (rather than just expected weight difference being bounded).

Condition 2: Given the weight function W, the scheduling policy π satisfies the following property:

There exists a constant $G < \infty$ such that the schedule weight of the scheduling policy π, $W^{\pi}(t) = W_{\mathbf{\Pi}(t)}(t)$, satisfies

$$W^{\pi}(t) \geq W^*(t) - G, \quad \forall t$$

Theorem 2: Given any reconfiguration delay $\Delta_r > 0$. Assume the arrival traffic is admissible and satisfies Assumption 1. Assume the weight function W satisfies Assumption 2' with function f. Suppose the scheduling policy π satisfies Condition 5 under the weight function W. If the sublinear function $g(\cdot)$ satisfies $\lim_{x \to \infty} \frac{f'(x)}{g(f(x))} = 0$, then the g-adaptive variant of π achieves throughput optimality.

The proof is given in Appendix C. Now the throughput optimality of the g-adaptive variant of MaxWeight-α ($\alpha > 0$) in example 1 is given by the next lemma following Theorem 2.

Corollary 4: Given $\alpha > 0$, suppose the sublinear function g satisfies $\lim_{x \to \infty} \frac{x^{\alpha - 1}}{g(x^{\alpha})} = 0$, then the g-adaptive variant of MaxWeight-α is throughput optimal.

IV. Discussion

In this section, we first give a brief explanation on how the adaptive policies achieve the throughput optimality. We then categorize policies dealing with nonzero reconfiguration delay into two classes, namely quasi-static and dynamic policies.

A. Queue Dynamics Under the Adaptive MaxWeight Policy

Fig. 2 illustrates the sample paths of the total queue length, the maximum weight, and the schedule weight of a network under the Adaptive MaxWeight policy. The green vertical lines mark the times of schedule reconfiguration, t_k^S, and the schedule weight is considered as zero during the reconfiguration delay (time period of length Δ_r following each schedule reconfiguration instance t_k^S). We first observe that the schedule duration increases as the total queue length increases. This is an essential component in achieving the network stability: let T be the mean schedule duration, then in order to ensure stability under $\Delta_r > 0$, the rate of schedule reconfiguration must satisfy $1 - \frac{\Delta_r}{T} > \rho$. The schedule duration increases with the queue length until this condition is satisfied.

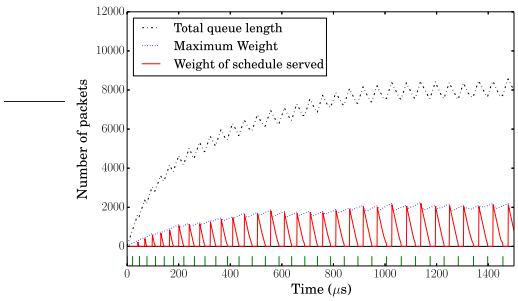

Fig. 2. Sample paths of the total queue length, maximum weight, and the schedule weight, under the Adaptive MaxWeight (AMW) policy. The number of ToRs is $N = 8$, the traffic load is $\rho = 0.6$, and the parameters of the AMW policy are $\delta = 0.01, \gamma = 0.1$. The green lines mark the schedule reconfiguration time instances t_k^S.

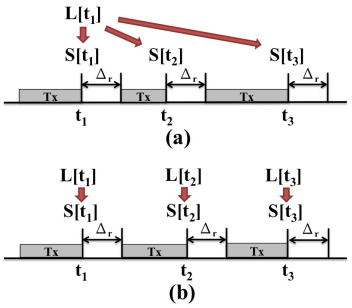

Fig. 3. Timing diagram for different scheduling strategies. (a) Quasi-static policies: Series of schedules determined in a single schedule computation, and some schedules could depend on out-dated queue information when being deployed. (b) Dynamic policies: Each schedule is computed based on the most up-to-date edge queue information.

A somewhat more interesting observation is in the mechanism of this schedule duration adjustment. Recall that in the construction of g-adaptive policy, it requires no explicit adjustment of the schedule duration, *i.e.* the duration of a schedule is not explicitly set at the time it is reconfigured. Instead, through the schedule determination by weight comparison at each time slot, the schedule duration is implicitly "adapted" to the appropriate value. After each schedule reconfiguration, the schedule weight begins from the maximum weight and decreases as the network is serving the queues associate with the schedule. When the total queue length is larger (and accordingly the maximum weight is larger), the threshold is larger and it takes longer for the weight difference between the maximum weight and the schedule weight to surpass the threshold. This property may be characterized by lemma 1 shown in Appendix A. It is worth mentioning that a similar mechanism also occurs in the Adaptive Back-pressure [10] proposed under the context of single-commodity, multi-hop networks with reconfiguration delay. While their network setting is different from this work in the sense that nodes in the network could perform reconfiguration asynchronously and require in-network buffers, each node exhibit the behavior of slowing down schedule reconfigurations as buffers piling up.

Note that this adaptive mechanism differs from other scheduling policies in the literature that explicitly adjust the schedule duration ([4]–[6]). We give a brief introduction of these policies and categorize them based on this schedule provisioning behavior in the next subsection. The performance comparison of these policies is then evaluated through simulations and presented in the next section.

B. Quasi-static v.s. Dynamic Policies

For scheduling policies accounting for nonzero reconfiguration delay, we classify them into two categories: "quasi-static scheduling" and "dynamic scheduling." Quasi-static scheduling policies, also refered to as "batch scheduling," [5] select a series of schedules based on a single schedule computation process, as shown in Figure 3 (a). We argue that under these policies, the generated schedules may depend on very out-dated information, especially for schedules employed later in a batch. This is the source of significant performance degradation. In contrast, under dynamic scheduling policies, each schedule is generated based on the most up-to-date queue information, as shown in Figure 3 (b). We now describe several example policies for each class and discuss their performance.

In [5], batch scheduling policies are further classified as fixed batch scheduling (FBS) policies or adaptive batch scheduling (ABS) policies depending on how to determine the time instances for schedule computation. If the time between two schedule computation is a fixed duration then it is a FBS policy, otherwise it is a ABS policy. In general, a FBS policy does not guarantee stability unless the number of schedules employed within a batch is restricted to avoid too frequent schedule reconfiguration. The traffic matrix scheduling (TMS) policy [4] is a special case of FBS policy which could guarantee stability if the traffic load is known in advance. Under the TMS policy, the schedule computation occurs at $t_k = kW$ for some integer parameter $W < \infty$ and $k = 0, 1, \ldots$ The TMS policy utilizes the Birkhoff von-Neumann (BvN) decomposition [22] to determine the schedules in the following W time slots. [2] More specifically, the scheduler takes the queue length information $\mathbf{L}(t_k)$ and scales it to a doubly stochastic matrix $\mathbf{B}(t_k)$ which indicates the relative service requirement in the following W slots and performs the BvN decomposition on $\mathbf{B}(t_k)$ as $\mathbf{B}(t_k) = \sum_{i=1}^{Q} \alpha_i \mathbf{P}_i$. Note the number of terms Q in the decomposition may vary (while $Q \leq N^2 - 2N + 2$). In practice, the parameter Q is set as a fixed number to avoid excessive schedule changes and Q largest weighted schedules are chosen. If the arrival traffic load is known a priori, an appropriate choice of parameters ensures the stability of TMS. Besides the TMS policy, other scheduling policies that determines schedules based on decomposition of the workload (*e.g.* [24]–[26]) could also be considered as variants of the FBS policy.

The ABS policy proposed in [5] determines each schedule computation time as the time the packets from last batch are cleared. The ABS guarantees rate stability [5] which is a weaker notion of stability compared to the strong stability considered in this paper. In fact, the expected queue length may be unbounded under the ABS policy, hence it is not considered in the performance comparison in this work. On the other hand, the Variable Frame MaxWeight (VFMW) policy

[2]The BvN decomposition is based on the BvN Theorem [23] stating that every doubly stochastic matrix could be decomposed as a convex combination of permutation matrices

proposed in [6] can be viewed as a variant to the ABS policy. The VFMW policy selects only one schedule for each batch, and the batch duration is a function of the batch size (instead of being the time that the batch is cleared). While the VFMW policy is shown in [6] to be throughput optimal, it has poor delay performance since it selects and fixes the schedule duration disregarding the arrivals in the schedule duration, which is a similar problem to other quasi-static policies.

In contrast to the quasi-static policies, under the dynamic scheduling policies, the schedule being employed at each schedule reconfiguration instance is based on the most up-to-date queue length information. Frame-based policies, such as the Fixed Frame MaxWeight (FFMW) policy [7], are examples for dynamic policies. The FFMW policy selects a fixed period T and sets the schedule reconfiguration times at $t_n = nT, n = 0, 1, 2, \ldots$ At each t_n, the schedule is selected as the MaxWeight schedule at time t_n, therefore each schedule is based on the most up-to-date queue information and result in an improved delay performance. Unfortunately, like the TMS policy, however, the FFMW requires prior knowledge of the traffic load to guarantee the network stability which is similar to the TMS policy. Note that by the definition given, our proposed g-adaptive policies are other examples of dynamic policies. In contrast to the FFMW, however, our proposed g-adaptive policies achieve throughput optimality without prior knowledge of the traffic load. Moreover, the g-adaptive policies also have good delay performance, as would be shown through simulations in the next section.

V. Simulation

In this section we present simulation results for the AMW policy in Example 1, and compare them to the benchmark scheduling policies such as TMS, FFMW and VFMW. We also present comparison of the AMW policy against adaptive variants of lower complexity policies approximating the MaxWeight policy, as given in the examples in section III-A.

The experiments are conducted with the simulator built for the REACToR switch in [1]. The reconfiguration delay is $\Delta_r = 20\ \mu s$. In order to compare scheduling policies in optical switches, we cease the electronic switches in the hybrid switch in [1] and only utilize the optical switches. We consider $N = 100$ ToR switches, and the network topology is assumed to be non-blocking. Therefore, the set of feasible schedules $\mathcal{S}$ is in fact the set of $N \times N$ permutation matrices. Each link has data bandwidth $B = 100$ Gbps, and the packets are of the same size $p = 1500$ bytes (each takes $0.12\mu s$ for transmission). Each edge queue can store up to 1×10^5 packets, and incoming packets are discarded when the queue is full.

The traffic is assumed to be admissible, i.e. $\rho(\boldsymbol{\lambda}) < 1$, while the load matrices $\boldsymbol{\lambda}$ are classified as the following types:

1) Uniform: $\lambda_{ij} = \rho/N,\ \forall 1 \leq i, j \leq N$.
2) Nonuniform: $\lambda_{ij} = \frac{\rho}{M} \sum_{m=1}^{M} \mathbf{P}_{ij}^m$ where $\mathbf{P}^m \in \mathcal{P}$ are randomly selected permutation matrices, and $M = 100$.

The performance measure used is the mean edge queue length (averaged over queues and over time). Notice that the expected average delay of the entire network is linearly related to this quantity according to the Little's law.

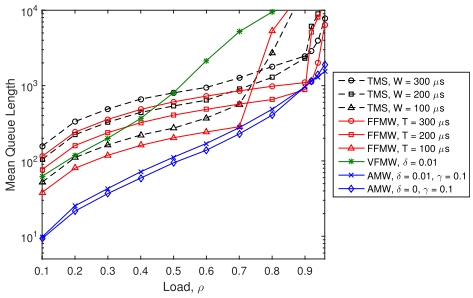

Fig. 4. Mean queue length versus traffic load ρ under **uniform traffic**. The TMS policy reconfigures the schedule $Q = 10$ times within QT slots. The scheduling rate under either the TMS or PMW is equal to $1/T$, while under AMW is adapted to the traffic load intensity.

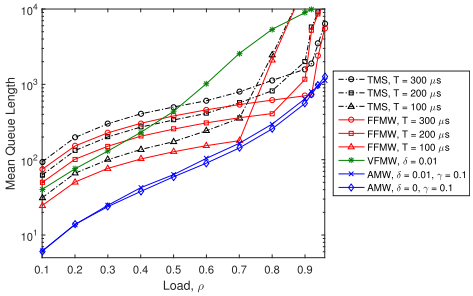

Fig. 5. Mean queue length versus traffic load ρ under **nonuniform traffic**. The TMS policy reconfigures the schedule $Q = 10$ times within QT slots.

In Figs 4 and 5, we compare the scheduling policies described in section IV under the uniform and the nonuniform traffic, respectively. For TMS, we set the number of schedules used between two schedule computation time instances to be $Q = 10$. In Figs. 4 and 5 we can see that the TMS and FFMW perform comparably, while the FFMW slightly outperforms the TMS under the same schedule reconfiguration rate $1/T = 10/W$. We note that under both the TMS and FFMW policies, the traffic loads they could stabilize are determined by the reconfiguration rate $1/T = 10/W$. In general, a smaller T (W) value gives better delay performance at a fixed stable load, but choosing a smaller T (W) value also decreases the maximum load that the TMS or FFMW policy could stabilize, which makes the design choice highly dependent on the accuracy of the traffic load. We also note that the VFMW policy, while is known to achieve the throughput optimality, underperforms both the FFMW and TMS policies in practice. In contrast, the AMW policy shows an improved performance over the FFMW and TMS policies, and at the same time guarantees throughput optimality. Another important implication from the two figures is that even if we know exactly the traffic load ρ and fine tune the optimal schedule reconfiguration period T for the FFMW and TMS policies, they are at most comparable to the AMW policy. The AMW policy has the merit that achieves better performance (over other policies) at any traffic load without the need to adjust the parameters.

In the introduction (section I), we mentioned that the reconfiguration delay Δ_r depends on the optical switch technology chosen, as well as additional delay occurring at the controller and the end host/ToR. The end-to-end per packet delay of

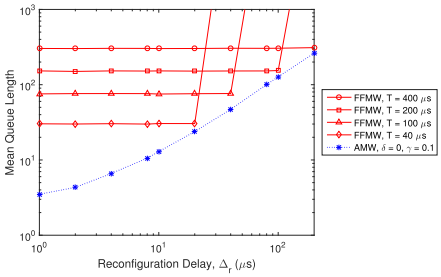

Fig. 6. Mean queue length versus the reconfiguration delay Δ_r under nonuniform traffic. The traffic load is fixed as $\rho = 0.3$.

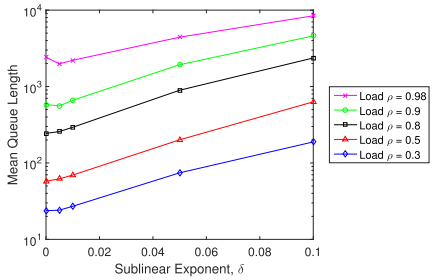

Fig. 7. Mean queue length versus sublinear exponent δ for the AMW policy. The threshold ratio is fixed as $\gamma = 0.01$.

the FFMW and AMW policies under various reconfiguration delay Δ_r are shown in Fig. 6. The traffic load is fixed as $\rho = 0.3$. We can see that the AMW outperforms the FFMW under each Δ_r value, regardless of the parameter selection of the FFMW policy. Even more notable, the performance of the AMW actually traces the optimal performance of the FFMW. This observation, along with the observation in the previous paragraph, suggests that the adaptive strategy of the AMW allows it to capture the optimal schedule reconfiguration rate based solely on the queue lengths and no prior knowledge of the arrival statistics is required.

The previous simulation results may also be used to evaluate the practicality of the proposed all-optical network architecture, in terms of the memory required at the ToR switches. For example, consider a network with $N = 100$ ToRs with reconfiguration delay $\Delta_r = 20$ μs as shown in Fig. 5. Even at a very high traffic load $\rho = 0.9$, the mean total queue length in a ToR (aggregating queues destined for all ToRs) is approximately $10^3 * 1500$ bytes $* N = 150$MBytes, which is still feasible in commercially available ToR switches [27].

We now consider the effect of the parameter selection for the AMW policy from example 1. In particular, we consider the performance as a function of parameters of the hysteresis function $g(x) = (1-\gamma)x^{1-\delta}$, evaluated under various traffic loads ($\rho \in \{0.3, 0.5, 0.8, 0.9, 0.98\}$). In Fig. 7, we fix the threshold ratio as $\gamma = 0.01$ and evaluate the performance under various sublinear exponent δ. Note that the mean queue length becomes shorter (better performance) when δ approaches 0 (which means g is closer to a linear function), and that the mean queue length increases quickly as δ becomes larger. This implies that we should avoid selecting $g(x)$ that grows too slow with x (such as large δ, or even logarithmic functions like $g(x) = \log(1+x)$) for better performance, despite the

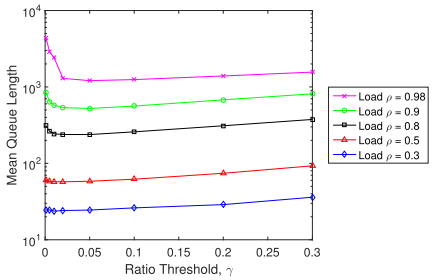

Fig. 8. Mean queue length versus threshold ratio γ for the AMW policy. The sublinear exponent is fixed as $\delta = 0$.

TABLE I
EXCESS OVER MINIMUM UNDER VARIOUS N, WITH $\Delta_r = 20\mu$S

N	4	8	16	32	64	100
Excess (%)	2.8	1.2	0.4	1.8	1.0	2.4

fact that throughput optimality guarantee from Section III actually applies for a large class of sublinear functions $g(x)$. Another observation we could make from Fig. 7 regards the performance at $\delta = 0$. Note that the throughput optimality guarantee does not include this case since g becomes a linear function. However, numerically and anecdotally the performance at $\delta = 0$ is generally good, and is even the optimal at most traffic loads (except at very high traffic load). The above observations simplify the parameter selection process: Since we rarely operate the network at traffic load close to the capacity, it is reasonable just to fix δ at a value close to 0.

In Fig. 8, we show the performance under variousvalues of ratio threshold γ, while the sublinear exponent is fixed as $\delta = 0$. Comparing with Fig. 7, we first note that the effect of γ on the performance is even less significant than the effect of δ. Moreover, from the curves we could find a region of γ values (around $[0.05, 0.1]$) where the performances are close to the optimal performance at each traffic load.

Combining the above results, the non-asymptotic delay performance of the AMW policy is quite robust to the parameter selection in the sense that we could easily select a combination of (δ, γ) that has comparable performance to the optimal, even if we have no knowledge of the traffic load. This result further strengthens our analytical result that the AMW policy achieves throughput optimality without any knowledge of the traffic load. Besides the traffic load, we also consider the robustness of parameter selection under different system parameters such as number of ToR switches N, and reconfiguration delay Δ_r. In Tables I and II, we compare the mean queue length under a set of fixed parameters $(\delta, \gamma) = (0.01, 0.05)$ to the minimum mean queue length (over all parameter combinations) under different system parameters, and compute the percentage of excess queue length. The traffic load is fixed as $\rho = 0.5$. We could see from the tables that the performance of the fixed parameter is close to the optimal in all these cases, and conclude that the robustness of the AMW also holds under different system parameters.

One of the shortcomings of the AMW policy is the complexity of computing the MaxWeight schedules. Next, we consider the adaptive variants of lower complexity scheduling

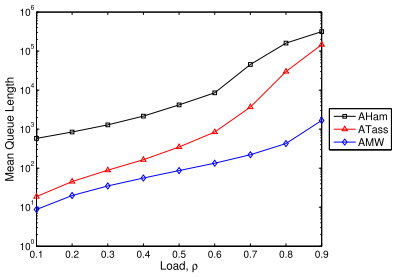

Fig. 9. Mean queue length versus traffic load for different adaptive policies. Number of ToRs is $N = 8$ and $g(x) = (1-\gamma)x^{1-\delta}$ with $\gamma = 0.1, \delta = 0.01$.

TABLE II

EXCESS OVER MINIMUM UNDER VARIOUS Δ_r, WITH $N = 100$

Δ_r (μs)	20	50	100	200	500
Excess (%)	2.4	3.1	2.4	2.1	1.6

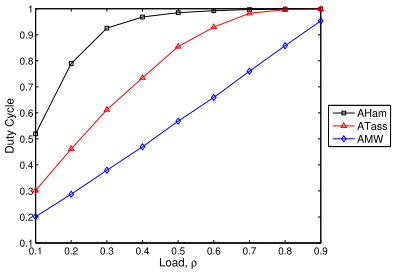

Fig. 10. Duty cycle versus traffic load for different adaptive policies. Number of ToRs is $N = 8$ and $g(x) = (1-\gamma)x^{1-\delta}$ with $\gamma = 0.1, \delta = 0.01$.

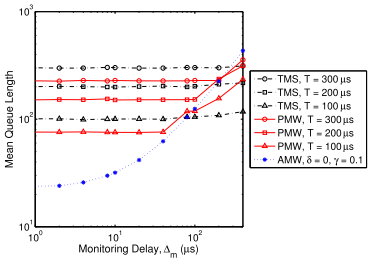

Fig. 11. Mean queue length versus monitoring delay Δ_m under nonuniform traffic. The traffic load is fixed as $\rho = 0.3$. The edge queue state is monitored/updated every microsecond.

policies introduced in examples 6 and 1. Fig. 9 shows the mean queue length versus traffic load for the Adaptive Hamiltonian (AHam), the Adaptive Tassiulas random policy (ATass), and the AMW policies, under uniform traffic. Since the delay performance of the lower complexity policies degrade drastically at large number of ToRs ($G \approx N!$ in Condition 1 for AHam and ATass), hence we set $N = 8$ for the simulations here. We also show the duty cycles of these policies in Fig. 10, where the duty cycle is defined as $DC \triangleq 1 - \frac{\Delta_r}{\mathbb{E}[T]}$, and $\mathbb{E}[T]$ is the mean schedule duration. Note that a necessary condition for a scheduling policy to be throughput optimal under traffic load ρ is to satisfy $DC > \rho$, which is satisfied by all the scheduling policies shown here.

From Fig. 9, we could see that the delay performance for the lower complexity policies are worse than the AMW policy. We may also observe that as the traffic load gets larger, the performance difference increases. This is because the schedule weight decreases in a slower rate (due to higher arrival rate), and it takes more time for the lower complexity policies to find a schedule with high enough weight. This could be validated by Fig. 10 that the schedule durations become significantly long ($DC \to 1$) at high traffic loads. Note that the AHam policy has the worst delay performance for all traffic loads, this is probably because the Hamiltonian walk only changes few served queues in a time slot, and generally takes longer to find the next good schedule.

VI. CONCLUSION

This paper considers the end-to-end scheduling problem in all-optical data center networks. The entire network is viewed as a generalized crossbar interconnect with nonzero schedule reconfiguration delay. Due to the schedule reconfiguration delay, many throughput optimal scheduling policies (under zero reconfiguration delay) in the literature could not be directly applied in this problem.

In this paper, we propose a general method to develop a class of scheduling policies for scheduling with nonzero reconfiguration delay, namely the g-adaptive variant policies: Given any Markov policy π, a weight function W, and a sublinear hysteresis function $g(\cdot)$, we construct a g-adaptive variant of π which involves a weight comparison between the current schedule and the schedule generated by π, and reconfigure to the schedule generated by π when it is "significantly better." We show the throughput optimality of the g-adaptive variants of π given the weight of schedule generated by π is guaranteed to have bounded weight difference to the maximum weight policy (either in the deterministic or the expected sense).

Besides the per-packet delay performance, another performance measure for data center applications is the flow completion time, which addresses the time required to complete the transfer of batch of packets. Intuitively, while MaxWeight-like policies address the throughput optimality, they may suffer longer flow completion time in extreme cases such as a very long flow competing with short flows, since MaxWeight-like policies tend to prioritize long flows over short flows. Extensions of the Adaptive MaxWeight to also address the challenge of minimizing flow completion time would be an interesting future direction.

Another possible line of future research is to consider the effect of monitoring and computation delay in addition to the reconfiguration delay. In practice, monitoring (collecting queue length information) and schedule computation may also take a non-negligible time (denoted as Δ_m and Δ_c, respectively), as suggested in [8]. These delays can be viewed as incurring delay on the queue length information as in the Pipelined MaxWeight in example 2. In other word, the throughput optimality of a g-adaptive policy is ensured even for $\Delta_m, \Delta_c > 0$. However, our conclusion regarding the improved performance of dynamic policies over quasi-static policies does not hold when Δ_m, Δ_c increase significantly. As an example, Fig. 11 shows the delay performance under various Δ_m. In the small Δ_m regime, the AMW policy achieves substantially better performance over the comparing scheduling policies.

However, as Δ_m increases, the performance of the AMW policy sees a significant degradation. This observation motivates the following future research directions: 1) the development of low-delay ToR monitoring system [28], 2) lower complexity scheduling policies with comparable delay performance, and 3) improvements to the AMW policy in order to increase the robustness towards the monitoring and computation delay.

APPENDIX

The following Foster Lyapunov Theorem is used to show the stability in the proofs listed in this appendix.

Fact 1 (Foster-Lyapunov [29]): Given a system of edge queues $Q_{ij}, 1 \le i,j \le N$, with queue lengths $\mathbf{L}(t) = [L_{ij}(t)]$, which could be described by an irreducible discrete-time Markov chain (DTMC) on a countable state space $\mathcal{L}$. Let f, g be two nonnegative functions on $\mathcal{L}$ such that $\forall \mathbf{L}(t_k) \in \mathcal{L}$,

$$\mathbb{E}[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k))|\mathbf{L}(t_k)] \le -f\big(\mathbf{L}(t_k)\big) + g\big(\mathbf{L}(t_k)\big)$$

and suppose for some $\epsilon > 0$, the set $C = \{\mathbf{L} \in \mathcal{L} : f(\mathbf{L}) < g(\mathbf{L}) + \epsilon\}$ is finite, then the DTMC describing the queue length evolution is positive recurrent and we have

$$\lim_{k\to\infty} \mathbb{E}\left[f\big(\mathbf{L}(t_k)\big)\right] \le \lim_{k\to\infty} \mathbb{E}\left[g\big(\mathbf{L}(t_k)\big)\right]$$

A. Proof of Theorem 1

Notice that in the $\Delta_r > 0$ regime, the queue dynamics depends on whether the system is in reconfiguration. We use $r(t)$ to denote the remaining time for the system to be in reconfiguration at time t, while $r(t) = 0$ indicates that the system is not in reconfiguration.

Now define $Y(t) = \big(\mathbf{L}(t), \mathbf{S}(t-1), r(t)\big)$, then $\{Y(t)\}_{t=0}^{\infty}$ forms an irreducible DTMC. In the following, we determine the T-step conditional Lyapunov drift (where $T < \infty$ is to be determined later) to show the stability. In other words, we determine the conditional Lyapunov drift for the sampled process $\{Y(t_k)\}_{k=0}^{\infty}$ (which is also a DTMC) at times $t_k = kT$:

$$\begin{aligned}&\mathbb{E}^{\pi^g}\left[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k))\middle|Y(t_k)\right]\\ &\quad= \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}^{\pi^g}\left[\mathbb{E}^{\pi^g}\left[\Delta V(t)|Y(t)\right]\middle|Y(t_k)\right] \qquad (5)\end{aligned}$$

By the assumption of the Theorem and that the system could not serve the queues during reconfiguration, we may write

$$\begin{aligned}&\mathbb{E}^{\pi^g}\big[\Delta V(t)\big|Y(t)\big]\\ &\le \Lambda(\mathbf{L}(t)) - \mathbb{E}^{\pi^g}\big[W^{\pi^g}(t)\mathbb{1}_{\{r(t)=0\}}\big|Y(t)\big]\\ &\le -\epsilon W^*(t) + K\\ &\quad + \Big(W^*(t) - \mathbb{E}^{\pi^g}\big[W^{\pi^g}(t)\mathbb{1}_{\{r(t)=0\}}\big|Y(t)\big]\Big)\\ &\le -\epsilon W^*(t) + K + \Big(W^*(t) - \mathbb{E}^{\pi^g}\left[W^{\pi^g}(t)\middle|Y(t)\right]\Big)\\ &\quad + \mathbb{E}^{\pi^g}\left[W^*(t)\mathbb{1}_{\{r(t)>0\}}\middle|Y(t)\right], \qquad (6)\end{aligned}$$

where the last inequality follows from $W^{\pi^g}(t) \le W^*(t)$ and $\mathbb{1}_{\{r(t)=0\}} + \mathbb{1}_{\{r(t)>0\}} = 1$.

By the construction of the g-adaptive variant and that π satisfies Condition 1, we have

$$\begin{aligned}&W^*(t) - \mathbb{E}^{\pi^g}\left[W^{\pi^g}(t)\middle|Y(t)\right]\\ &\quad= W^*(t) - \mathbb{E}^{\pi}\left[W^{\pi}(t)\middle|Y(t)\right]\\ &\qquad + \mathbb{E}^{\pi^g}\left[W^{\pi}(t) - W^{\pi^g}(t)\middle|Y(t)\right]\\ &\quad\le G + \mathbb{E}^{\pi^g}\left[g(W^{\pi}(t))\middle|Y(t)\right] \le G + g(W^*(t)) \qquad (7)\end{aligned}$$

Since at most one packet could depart and at most $A_{\max}$ packets could arrive at each queue in a time slot, then for any $t \in [t_k, t_{k+1}]$, we have $|\mathbf{L}_{ij}(t) - \mathbf{L}_{ij}(t_k)| \le A_{\max}(t - t_k) \le A_{\max}T$, and thus $||\mathbf{L}(t) - \mathbf{L}(t_k)|| \le N^2 A_{\max} T$. Hence for any schedule $\mathbf{S}$, we have $|W_{\mathbf{S}}(\mathbf{L}(t)) - W_{\mathbf{S}}(\mathbf{L}(t_k))| \le BA_{\max}N^2T$, where B is the Lipshitz constant. With this relation we derive the following bounds for the maximum weight at time t:

$$\begin{aligned}W^*(t) = W_{\mathbf{\Pi}^*(t)}(\mathbf{L}(t)) &\le W_{\mathbf{\Pi}^*(t)}(\mathbf{L}(t_k)) + BA_{\max}N^2T\\ &\le W^*(t_k) + BA_{\max}N^2T\\ W^*(t) &\ge W_{\mathbf{\Pi}^*(t_k)}(\mathbf{L}(t))\\ &\ge W_{\mathbf{\Pi}^*(t_k)}(\mathbf{L}(t_k)) - BA_{\max}N^2T\\ &= W^*(t_k) - BA_{\max}N^2T \qquad (8)\end{aligned}$$

Apply (6)-(8) into (5), we obtain

$$\begin{aligned}&\mathbb{E}^{\pi^g}\left[V\left(\mathbf{L}(t_{k+1})\right) - V(\mathbf{L}(t_k))\middle|\mathbf{L}(t_k)\right]\\ &\le -\epsilon T(W^*(t_k) - BA_{\max}N^2T)\\ &\quad + Tg(W^*(t_k) + BA_{\max}N^2T)\\ &\quad + T(G+K) + \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}^{\pi^g}\left[W^*(t)\mathbb{1}_{\{r(t)>0\}}\middle|\mathbf{L}(t_k)\right]\end{aligned} \qquad (9)$$

In order to achieve the stability, it is necessary to ensure that the reconfiguration does not happen too often. We make this statement formal with the following lemma:

Lemma 1: Given any fixed $T > 0$ and a weight function W satisfying Assumption 2 with Lipschitz constant B. Let $g(x)$ be a sublinear and strictly increasing function, and let $h(x) = g(x - BA_{\max}N^2T) - 2BA_{\max}N^2T$. Suppose the Markov policy π satisfies Condition 1 under the weight function W, then under the g-adaptive variant of π:

$$\mathbb{E}^{\pi^g}\left[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{1}_{\{r(t)>0\}}\middle|Y(t_k)\right] \le \Delta_r + \frac{TG}{h(W^*(t_k))}$$

The proof of Lemma 1 is given in Appendix B. Note that lemma 1 gives an upper bound on the frequency of schedule reconfiguration when the queue length is large. Now select T such that $T > \frac{\Delta_r}{\epsilon}$, and apply Lemma 1 to (9), we obtain

$$\begin{aligned}&\mathbb{E}^{\pi^g}\left[V\left(\mathbf{L}(t_{k+1})\right) - V(\mathbf{L}(t_k))\middle|Y(t_k)\right]\\ &\quad\le -T\big(\epsilon - \frac{\Delta_r}{T} - \frac{G}{h(W^*(t_k))}\big)W^*(t_k)\\ &\qquad + Tg\big(W^*(t_k) + BA_{\max}N^2T\big) + T(G+K)\\ &\qquad + T\big(\epsilon + \frac{\Delta_r}{T} + \frac{G}{h(W^*(t_k))}\big)BA_{\max}N^2T \qquad (10)\end{aligned}$$

Since $T > \frac{\Delta_r}{\epsilon}$, we may select an arbitrary constant $\epsilon' \in (0, \epsilon - \frac{\Delta_r}{T})$. Then from the sublinearity of g, we obtain

$$\mathbb{E}^{\pi^g}\left[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k))\Big|Y(t_k)\right] < -T\epsilon' W^*(t_k) + K'$$

for some constant $K' < \infty$. It then follows Fact 1 that $\lim_{k\to\infty} W^*(t_k) < \infty$, and hence $\lim_{k\to\infty} ||\mathbf{L}(t_k)|| < \infty$, which then implies the strong stability. Since the derived strong stability holds for any admissible traffic load, we conclude that the g-adaptive variant of π achieves throughput optimality.

B. Proof of Lemma 1

Proof: Let $Z_{[s,t]}$ be the number of schedule reconfigurations within the time period $[s,t]$. In the following, we first derive an upper bound on $\Pr\{Z_{[t_k,t_{k+1}]} \le 1\}$, then use this bound to get an upper bound on the expected length of reconfiguration during $[t_k, t_{k+1}]$.

Let $t'_k = \min_{k: t_k^S \ge t} t_k^S$ denote the first reconfiguration instance after time t_k. Since either there is no reconfiguration within $[t_k, t_{k+1}]$ or $t'_k = \tau$ for some $\tau \in [t_k, t_{k+1}]$, we have

$$\Pr\left\{Z_{[t_k,t_{k+1}]} = 0\right\} + \sum_{\tau=t_k}^{t_{k+1}-1} \Pr\{\tau = t'_k\} = 1 \qquad (11)$$

We may also write the probability that exactly one schedule reconfiguration occurs in the interval $[t_k, t_{k+1}]$ as:

$$\begin{aligned}
&\Pr\{Z_{[t_k,t_{k+1}]} = 1\} \\
&= \sum_{\tau=t_k}^{t_{k+1}-1} \Pr\left\{Z_{[\tau,t_{k+1}]} = 0, \tau = t'_k\right\} \\
&= \sum_{\tau=t_k}^{t_{k+1}-1} \Pr\left\{Z_{[\tau,t_{k+1}]} = 0 \Big| \tau = t'_k\right\} \Pr\{\tau = t'_k\} \qquad (12)
\end{aligned}$$

We now show that if at the schedule reconfiguration instance τ, the weight difference is small, then no reconfiguration could occur in the interval $[\tau, t_{k+1}]$. In other words, we need to show that $W^\pi(\tau') - W_{\Pi(\tau)}(\tau') \le g(W^\pi(\tau'))$ for any $\tau' \in [\tau, t_{k+1}]$.

Let $h(x) = g(x - BA_{\max}N^2T - G) - 2BA_{\max}N^2T$, then if $W^*(\tau) - W_{\Pi^g(\tau)}(\tau) < h(W^*(t_k))$, we have:

$$\begin{aligned}
W^*(\tau') - W_{\Pi(\tau)}(\tau') &\le W^*(\tau') - W_{\Pi(\tau)}(\tau') \\
&\le W^*(\tau) - W_{\Pi(\tau)}(\tau) + 2BA_{\max}N^2(\tau' - \tau) \\
&< g(W^*(t_k) - BA_{\max}N^2T - G) \overset{(a)}{\le} g(W^\pi(\tau'))
\end{aligned}$$

where (a) follows from $W^\pi(\tau') \ge W^*(\tau') - G \ge W^*(t_k) - G - BA_{\max}N^2T$ and g strictly increasing. We thus have:

$$\begin{aligned}
&\Pr\left\{Z_{[\tau,t_{k+1}]} = 0 \Big| \tau = t'_k\right\} \\
&\ge \Pr\left\{W^*(\tau) - W_{\Pi(\tau)}(\tau) < h(W^*(t)) \Big| \tau = t'_k\right\} \\
&\overset{(b)}{\ge} 1 - \frac{\mathbb{E}^\pi\left[W^*(\tau) - W_{\Pi(\tau)}(\tau)\right]}{h(W^*(t_k))} \overset{(c)}{\ge} 1 - \frac{G}{h(W^*(t_k))}
\end{aligned} \qquad (13)$$

where (b) follows from the Markov's inequality, and (c) follows from Condition 1.

Hence by (12) and (13) we have that

$$\begin{aligned}
&\Pr\left\{Z_{[t_k,t_{k+1}]} \le 1\right\} \\
&\ge \Pr\left\{Z_{[t_k,t_{k+1}]} = 0\right\} \\
&\quad + \sum_{\tau=t_k}^{t_{k+1}-1} \left(1 - \frac{G}{h(W^*(t_k))}\right) \Pr\{\tau = t'_k\} \\
&\ge 1 - \frac{G}{h(W^*(t_k))} \qquad (14)
\end{aligned}$$

With the following bound which is obvious by definition:

$$\sum_{t=t_k}^{t_{k+1}-1} \mathbb{1}_{\{r(t)>0\}} \le \begin{cases} \Delta_r, & \text{if } Z_{[t_k,t_{k+1}]} \le 1 \\ T, & \text{if } Z_{[t_k,t_{k+1}]} > 1, \end{cases}$$

along with (14), we then have the bound on the expected schedule reconfiguration delay within the interval $[t_k, t_{k+1}]$:

$$\mathbb{E}^{\pi^g}\left[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{1}_{\{r(t)>0\}} \Big| Y(t_k)\right] \le \Delta_r + \frac{TG}{h(W^*(t_k))}$$

■

C. Proof of Theorem 2

Proof: We evaluate the conditional Lyapunov drift for the sampled process $\{Y(t_k)\}_{k=0}^\infty$ following (5). Now consider the Lyapunov function $V(\mathbf{L}) = \sum_{i,j=1}^N F(L_{ij})$, where $F(x) = \int_0^x f(y)dy$ and let $\mathbf{\Delta}(t) = \mathbf{A}(t) - \mathbf{D}(t)$, then following the derivation similar to [21], we have the one-step expected Lyapunov drift given by

$$\begin{aligned}
&\mathbb{E}^{\pi^g}\left[\Delta V(t) \Big| Y(t)\right] \\
&= \mathbb{E}^{\pi^g}\Big[\sum_{i,j}\Big(F(L_{ij}(t) + \Delta_{ij}(t)) - F(L_{ij}(t))\Big)\Big|Y(t)\Big] \\
&\le \mathbb{E}^{\pi^g}\left[\left\langle f(\mathbf{L}(t)), \mathbf{\Delta}(t)\right\rangle \Big| Y(t)\right] \\
&\quad + \sum_{i,j} \frac{\Delta_{ij}(t)^2}{2} \max_{u\in[-1,A_{\max}]} \left|f'(L_{ij}(t) + u)\right| \qquad (15)
\end{aligned}$$

Now note that

$$\begin{aligned}
&\mathbb{E}^{\pi^g}\left[\left\langle f(\mathbf{L}(t)), \mathbf{\Delta}(t)\right\rangle \Big| Y(t)\right] \\
&= \mathbb{E}\left[\left\langle f(\mathbf{L}(t)), \mathbf{A}(t) - \mathbf{\Pi}^g(t)\mathbb{1}_{\{r(t)=0\}}\right\rangle \Big| Y(t)\right] \\
&= \left\langle f(\mathbf{L}(t)), \boldsymbol{\lambda} - \mathbf{\Pi}^g(t)\right\rangle + W^{\pi^g}(t)\mathbb{1}_{\{r(t)>0\}} \\
&\le \left\langle f(\mathbf{L}(t)), \boldsymbol{\lambda} - \mathbf{\Pi}^*(t)\right\rangle + \left(W^*(t) - W^{\pi^g}(t)\right) \\
&\quad + W^*(t)\mathbb{1}_{\{r(t)>0\}} \qquad (16)
\end{aligned}$$

By the definition of the traffic load ρ, we may write $\boldsymbol{\lambda} = \rho\sum_{i=1}^I \mathbf{P}_i$, where $\mathbf{P}_i \in \mathcal{S}$ for $i = 1, \ldots, I$. Then by the definition of the MaxWeight schedule, we have

$$\left\langle f(\mathbf{L}(t)), \boldsymbol{\lambda} - \mathbf{\Pi}^*(t)\right\rangle \le -(1-\rho)W^*(t) \qquad (17)$$

Also by the construction of the g-adaptive policy and the fact that the scheduling policy satisfies Condition 5, we have

$$\begin{aligned}
W^*(t) = W^{\pi^g}(t) &= [W^*(t) - W^\pi(t)] + [W^\pi(t) - W^{\pi^g}(t)] \\
&\le G + g(W^\pi(t)) \le G + g(W^*(t)) \qquad (18)
\end{aligned}$$

By assumption 1, we have $\Delta_{ij}^2(t) \leq A_{\max}^2$. Also, by the assumption that $\lim_{x\to\infty} \frac{f'(x)}{g(f(x))} = 0$, we have that $\forall \epsilon > 0$, there exists a constant $K_\epsilon < \infty$ such that

$$\max_{u\in[-T,A_{\max}T]} \Big| f'\left(x+u\right) \Big| \leq \epsilon g(f(x)) + K_\epsilon. \tag{19}$$

We then obtain the following:

$$\begin{aligned}
&\sum_{ij} \frac{\Delta_{ij}^2(t)}{2} \max_{u\in[-1,A_{\max}]} \Big| f'\left(L_{ij}(t)+u\right) \Big| \\
&\quad \leq \frac{N^2A_{\max}^2}{2}\left(\epsilon g\left(f\left(\max_{i,j} L_{ij}(t)\right)\right) + K_\epsilon\right) \\
&\quad \leq \frac{N^2A_{\max}^2}{2}\Big(\epsilon g(W^*(t)) + K_\epsilon\Big)
\end{aligned} \tag{20}$$

We thus have

$$\begin{aligned}
&\mathbb{E}^{\pi^g}\Big[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k))\Big|Y(t_k)\Big] \\
&\leq \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}^{\pi^g}\Big[-(1-\rho)W^*(t) + W^*(t)\mathbb{1}_{\{r(t)>0\}} + g(W^*(t)) \\
&\quad + \frac{N^2A_{\max}^2}{2}(\epsilon g(W^*(t)) + K_\epsilon) + G\Big|Y(t_k)\Big]
\end{aligned} \tag{21}$$

We now justify that the reconfiguration overhead is upper bounded when the queue lengths are large. This is characterized by the following lemma:

Lemma 2: Given any fixed $T > 0$ and a scheduling policy π satisfying Condition 5 under the weight function $f(\cdot)$. Let $g(\cdot)$ be a sublinear, strictly increasing function satisfying $\lim_{x\to\infty} \frac{f'(x)}{g(f(x))} = 0$. Then there exists a constant $M < \infty$ such that, if a reconfiguration occurs at time t and $W^*(t_k) > M$, then no reconfiguration could occur in $[t+1, t+T]$.

We postpone the proof of Lemma 2 to Appendix D. Now by Lemma 2, if $W^*(t_k) > M$, then since at most one schedule reconfiguration could occur within $[t_k, t_{k+1}]$, we have

$$\sum_{t=t_k}^{t_{k+1}-1} \mathbb{1}_{\{r(t)>0\}} \leq \Delta_r \tag{22}$$

Similar to (8), we derive the following lower and upper bounds on the change of the maximum weight:

$$\begin{aligned}
&W^*(t) = W_{\Pi^*(t)}(\mathbf{L}(t)) \leq \sum_{i,j} \Pi_{ij}^*(t) f(L_{ij}(t_k) + A_{\max}T) \\
&\leq \sum_{i,j} \Pi_{ij}^*(t)\Big(f(L_{ij}(t_k)) \\
&\qquad + A_{\max}T \max_{u\in[0,A_{\max}T]} |f'(L_{ij}(t_k)+u)|\Big) \\
&\leq W^*(t_k) + NA_{\max}T\left(\epsilon g(W^*(t_k)) + K_\epsilon\right)
\end{aligned} \tag{23}$$

$$\begin{aligned}
&W^*(t) \geq W_{\Pi^*(t_k)}(\mathbf{L}(t)) \geq \sum_{i,j} \Pi_{ij}^*(t_k) f(L_{ij}(t_k) - T) \\
&\geq \sum_{i,j} \Pi_{ij}^*(t_k)\Big(f(L_{ij}(t_k)) - T \max_{u\in[-T,0]} |f'(L_{ij}(t_k)+u)|\Big) \\
&\geq W^*(t_k) - NT\left(\epsilon g(W^*(t_k)) + K_\epsilon\right)
\end{aligned} \tag{24}$$

Apply (22) - (24) into (21), then $\forall Y(t_k) : W^*(t_k) > M$,

$$\begin{aligned}
&\mathbb{E}^{\pi^g}\Big[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k))\Big|Y(t_k)\Big] \\
&\leq -T\big(1-\rho-\frac{\Delta_r}{T}\big)W^*(t_k) \\
&\quad + (T(1-\rho) + \Delta_r A_{\max})NT(\epsilon g(W^*(t_k)) + K_\epsilon) \\
&\quad + Tg(W^*(t_k) + NA_{\max}T(\epsilon g(W^*(t_k)) + K_\epsilon)) + TG \\
&\quad + \frac{T}{2}N^2A_{\max}^2\big(\epsilon g(W^*(t_k) \\
&\qquad + NA_{\max}T(\epsilon g(W^*(t_k)) + K_\epsilon)) + K_\epsilon\big)
\end{aligned}$$

Since $T > 0$ may be selected arbitrarily, we select $T > \frac{\Delta_r}{1-\rho}$ so that $1-\rho-\frac{\Delta_r}{T} > 0$. Then by Fact 1, we have that $\{Y(t_k)\}_{k=0}^\infty$ is positive recurrent under the g-adaptive variant of π, and the queue lengths satisfy $\lim_{k\to\infty} \mathbb{E}\left[f(||\mathbf{L}(t_k)||)\right] < \infty$.

■

D. Proof of Lemma 2

Proof: By the assumption, the schedule is reconfigured to $\mathbf{\Pi}(t)$ at time t, and according to Condition 5 we have $W^\pi(t) = W_{\mathbf{\Pi}(t)}(\mathbf{L}(t)) \geq W^*(t) - G$. It now suffices to show that at any time $\tau \in [t+1, t+T]$, the weight of $\mathbf{\Pi}(t)$ is large enough so that no schedule reconfiguration occurs.

First recall that at most one packet could depart from each queue in a time slot, and each schedule has at most N parallel connections. Along with (19), we obtain

$$\begin{aligned}
W_{\mathbf{\Pi}(t)}(\tau) &= \Big\langle f(\mathbf{L}(\tau)), \mathbf{\Pi}(t)\Big\rangle \\
&\geq W_{\mathbf{\Pi}(t)}(t) - NT\max_{i,j}\big(\max_{u\in[-T,0]} |f'\left(L_{ij}(t)+u\right)|\big) \\
&\geq W^*(t) - G - NT\max_{i,j}\left(\epsilon g(f(L_{ij}(t))) + K_\epsilon\right)
\end{aligned} \tag{25}$$

On the other hand, since the arrival at each queue is upper bounded by $A_{\max}$ according to assumption 1, we have an upper bound for the maximum weight given by

$$\begin{aligned}
&W^*(\tau) = W_{\mathbf{\Pi}^*(\tau)}(\tau) \\
&\leq W_{\mathbf{\Pi}^*(\tau)}(t) + NA_{\max}T\max_{i,j}\Big(\max_{u\in[0,A_{\max}T]} |f'(L_{ij}(t)+u)|\Big) \\
&\leq W^*(t) + NA_{\max}T\max_{i,j}\Big(\epsilon g(f(L_{ij}(t))) + K_\epsilon\Big)
\end{aligned} \tag{26}$$

Combining (25) and (26) we get an upper bound for the weight difference between $\mathbf{\Pi}(t)$ and $W^\pi(\tau)$:

$$\begin{aligned}
&W^\pi(\tau) - W_{\mathbf{\Pi}(t)}(\tau) \leq W^*(\tau) - W_{\mathbf{\Pi}(t)}(\tau) \\
&\quad \leq N(A_{\max}+1)T\Big(\epsilon g\Big(f\Big(\max_{i,j} L_{ij}(t)\Big)\Big) + K_\epsilon\Big) + G \\
&\quad \leq N(A_{\max}+1)T\left(\epsilon g\left(W^*(t)\right) + K_\epsilon\right) + G
\end{aligned} \tag{27}$$

Similar to (25), since at most one packet could depart from each queue in a time slot, and since $g(\cdot)$ is a strictly increasing function, we obtain a lower bound for the threshold:

$$\begin{aligned}
&g\left(W^\pi(\tau)\right) \geq g\left(W^*(\tau) - G\right) \\
&\quad \geq g\Big(W^*(t) - NT\max_{i,j}\Big(\max_{u\in[-T,0]}\Big|f'(L_{ij}(t)+u)\Big|\Big) - G\Big) \\
&\quad \geq g\Big(W^*(t) - NT(\epsilon g(W^*(t)) + K_\epsilon) - G\Big)
\end{aligned} \tag{28}$$

Select ϵ to be sufficiently small, then since $g(\cdot)$ is a sublinear function, there exists a constant $M < \infty$ such that the right

hand side of (28) is larger than the right hand side of (27) whenever $W^*(t) > M$. Therefore, if $W^*(t) > M$, then no reconfiguration occurs for any $\tau \in [t+1, t+T]$. ■

REFERENCES

[1] H. Liu *et al.*, "Circuit switching under the radar with reactor," in *Proc. 11th USENIX Conf. Netw. Syst. Design Implement.*, Berkeley, CA, USA, 2014, pp. 1–15.

[2] K. Thakulsukanant, "Mems technology for optical switching," *Walailak J. Sci. Technol.*, vol. 10, no. 1, pp. 9–18, 2013.

[3] N. Farrington *et al.*, "A multiport microsecond optical circuit switch for data center networking," *IEEE Photon. Technol. Lett.*, vol. 25, no. 16, pp. 1589–1592, Aug. 15, 2013.

[4] G. Porter *et al.*, "Integrating microsecond circuit switching into the data center," in *Proc. Conf. SIGCOMM*, New York, NY, USA, 2013, pp. 447–458.

[5] K. Ross and N. Bambos, "Adaptive batch scheduling for packet switching with delays," in *High-performance Packet Switching Architectures*, I. Elhanany and M. Hamdi, Eds. London, U.K.: Springer, 2007, pp. 65–79.

[6] G. D. Çelik and E. Modiano, "Scheduling in networks with time-varying channels and reconfiguration delay," *IEEE/ACM Trans. Netw.*, vol. 23, no. 1, pp. 99–113, Feb. 2015.

[7] Y. Li, S. Panwar, and H. Chao, "Frame-based matching algorithms for optical switches," in *Proc. High Perform. Switching Routing*, Jun. 2003, pp. 97–102.

[8] C.-H. Wang, T. Javidi, and G. Porter, "End-to-end scheduling for all-optical data centers," in *Proc. IEEE INFOCOM*, Apr. 2015, pp. 406–414.

[9] G. Celik, S. C. Borst, P. A. Whiting, and E. Modiano, "Dynamic scheduling with reconfiguration delays," *Queueing Syst.*, vol. 83, no. 1, pp. 1–43, 2016.

[10] L. Tassiulas, "Adaptive back-pressure congestion control based on local information," *IEEE Trans. Autom. Control*, vol. 40, no. 2, pp. 236–250, Feb. 1995.

[11] G. Wang *et al.*, "c-through: Part-time optics in data centers," in *Proc. ACM SIGCOMM*, Aug. 2010, pp. 327–338.

[12] N. Farrington *et al.*, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proc. ACM SIGCOMM*, Aug. 2010, pp. 339–350.

[13] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "A centralized 'zero-queue' datacenter network," in *Proc. ACM SIGCOMM*, Chicago, IL, USA, Aug. 2014, pp. 307–318.

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

[15] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proc. 17th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, vol. 2. Mar. 1998, pp. 533–539.

[16] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 4, pp. 546–559, May 2003.

[17] D. Shah and D. Wischik, "Optimal scheduling algorithms for input-queued switches," in *Proc. 25th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2006, pp. 1–11.

[18] D. Shah and M. Kopikare, "Delay bounds for approximate maximum weight matching algorithms for input queued switches," in *Proc. 21st Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, vol. 2. Jun. 2002, pp. 1024–1031.

[19] K. Ross and N. Bambos, "Projective cone scheduling (PCS) algorithms for packet switches of maximal throughput," *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 976–989, Jun. 2009.

[20] M. Naghshvar, H. Zhuang, and T. Javidi, "A general class of throughput optimal routing policies in multi-hop wireless networks," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2175–2193, Apr. 2012.

[21] I. Keslassy and N. McKeown, "Analysis of scheduling algorithms that provide 100% throughput in input-queued switches," in *Proc. Annu. Allerton Conf. Commun. Control Comput.*, Oct. 2001, pp. 593–602.

[22] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," in *Proc. 19th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, vol. 3. Mar. 2000, pp. 1614–1623.

[23] G. D. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucum´an Rev. Ser. A*, vol. 5, pp. 147–150, 1946.

[24] A. Dasylva and R. Srikant, "Optimal WDM schedules for optical star networks," *IEEE/ACM Trans. Netw.*, vol. 7, no. 3, pp. 446–456, Jun. 1999.

[25] X. Li and M. Hamdi, "On scheduling optical packet switches with reconfiguration delay," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 7, pp. 1156–1164, Sep. 2003.

[26] S. B. Venkatakrishnan, M. Alizadeh, and P. Viswanath, "Costly circuits, submodular schedules and approximate carathodory theorems," in *Proc. 13th ACM SIGMETRICS/PERFORMANCE Joint Int. Conf. Meas. Modeling Comput. Syst.*, 2016, pp. 75–88.

[27] *Why Big Data Needs Big Buffer Switches*, accessed on Oct. 13, 2016. [Online]. Available: https://www.arista.com/assets/data/pdf/Whitepapers/BigDataBigBuffers-WP%.pdf

[28] T. Aktaş, C.-H. Wang, and T. Javidi, "WiCOD: Wireless control plane serving an all-optical data center," in *Proc. 13th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, May 2015, pp. 299–306.

[29] B. Hajek, "Notes for ECE 534: An exploration of random processes for engineers," Univ. Illinois, Urbana-Champaign, IL, USA, 2011.

Chang-Heng Wang (S'14) received the B.S. degree in electrical engineering from National Taiwan University, Taiwan, in 2011. He is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, University of California at San Diego, San Diego, CA, USA. His research interests are in stochastic networks, network algorithms, and wireless communications.

Tara Javidi (S'96–M'02–SM'12) received the degree in electrical engineering from the Sharif University of Technology in 1996, the M.S. degrees in electrical engineering and computer science: systems and applied mathematics: stochastics from the University of Michigan, Ann Arbor, MI, USA, and the Ph.D. degree in EECS from the University of Michigan in 2002. From 2002 to 2004, she was an Assistant Professor with the Electrical Engineering Department, University of Washington, Seattle, WA, USA. She joined the University of California at San Diego, San Diego, CA, USA, in 2005, where she is currently a Professor of electrical and computer engineering. Her research interests are in communication networks, stochastic resource allocation, and wireless communications. She was a Barbour Scholar from 1999 to 2000 and received the NSF CAREER Award in 2004.