End-to-End Scheduling for All-Optical Data Centers

Chang-Heng Wang, Tara Javidi, and George Porter
University of California, San Diego
Email: {chw009, tjavidi}@ucsd.edu, gmporter@cs.ucsd.edu

***Abstract*—This paper considers the end-to-end scheduling for all-optical data center networks with zero in-network buffer and non-negligible reconfiguration delay. It is known that in the regime where the scheduling reconfiguration delay is non-negligible, the rate of schedule reconfiguration should be limited in such a way as to minimize the impact of reduced duty-cycles and to ensure bounded delay. However, when the scheduling rate is restricted, the existing literature also tends to restrict the rate of monitoring and decision processes. We first present a framework for scheduling with reconfiguration delay that decouples the rate of scheduling from the rate of monitoring. Under this framework, we then present two scheduling algorithms for switches with reconfiguration delay, both based on the well-known MaxWeight scheduling policy. The first one is the Periodic MaxWeight (PMW), which is simpler in computation, but requires prior knowledge of traffic load. The other is the Adaptive MaxWeight (AMW), which, in contrast, requires no prior knowledge. We show the stability condition for both algorithms and evaluate their delay performance through simulations.**

I. Introduction

An increasing array of online services and applications are provided online–from search and social networks, to entertainment and streaming video, to healthcare and government systems. Each of these applications relies on enormous amounts of data processing to provide useful content to the end user, and the underlying compute and storage infrastructure needed to support these applications are increasingly hosted in Internet data centers. Data centers may exhibit enormous scale–hosting hundreds of thousands of servers.

The high cost, power demand, and complexity hinder the adoption of the full bisection bandwidth topologies, such as FatTrees [1], in data center networks. Data center operators instead typically rely on *oversubscription* to reduce network cost and power by providing a reduced quantity of bisection network bandwith. The downside of oversubscription is poor application performance and poor server utilization, since servers have to wait for data to arrive over the congested network fabric. More recently, a number of researchers have proposed reconfigurable network topologies, such as switched optical pathways [2], [3], [4]. Reconfigurable network topologies offer very high bisection bandwidth but do not require several layers of network switches as in FatTrees.

The relatively low costs of reconfigurable optical network topologies make them promising candidates for data center networks, nevertheless, there are still two main challenges for the adoption of reconfigurable optical circuits. Firstly, since reconfigurable optical circuits are inherently bufferless, data must be buffered at the source before transmission. Bufferless circuit-based networks are fundamentally different from buffered packet-switched networks. Since data transmissions cannot rely on buffers along the path, the network control plane must ensure that data is ready to send along the end-to-end circuit, with buffering only at the edge of the network. This network topology can be viewed as a single crossbar interconnecting the top of rack (ToR) switches, except that the full bisection bandwidth is not guaranteed. Specifically, due to the topology constraint, there are certain circuit configurations that could not allow all the ToR switches to transmit at the same time.

Secondly, candidate optical circuit switching technologies (such as "binary MEMS" mirror arrays [5]) typically exhibit a *reconfiguration delay* when the circuit configuration is changed. This delay is a period where data cannot flow through the switch, and for practical circuit switch technologies, this reconfiguration delay is significantly longer than the link-layer inter-frame gap. For example, the reconfiguration delay for state of the art binary MEMS is $2 - 20$ μs [5], which is significantly larger than the interframe gap of 9.6 ns. This nonzero reconfiguration delay motivates the need for scheduling policies that account for the reconfiguration delay.

In this work, we propose scheduling policies that systematically address both challenges outlined above. The proposed solutions are based on the MaxWeight policy [6] extensively studied in the context of switching fabrics and crossbar switches. When specialized to an all-optical network, the MaxWeight policy configures the current schedule based on current queue backlog information at the Top of the Rack (ToR). Let the weight of a schedule to be defined as the weighted sum of the queue lengths. Under the MaxWeight policy, at each time slot, the scheduler computes the weight for each schedule and selects the schedule with maximum weight. In [7] and [8], the MaxWeight has been shown to stabilize any admissible packet arrival process. Our proposed Periodic MaxWeight (PMW) policy periodically reconfigures the optical circuit according to the MaxWeight schedule, while the period is selected appropriately given the prior knowledge of the traffic statistics. On the other hand, the Adaptive MaxWeight (AMW) policy selects the circuit reconfiguration time in an adaptive fashion and based on the effectiveness of the current schedule. We will show that the PMW policy could stabilize the network if the traffic load is given, while the AMW policy could stabilize any admissible traffic without any knowledge of the traffic statistics (hence, achieving 100% throughput).

The rest of the paper is organized as follows. In the next section, the network model as well as the timing parameters associated with the monitoring, computation, and reconfiguration processes are introduced. In section III, we briefly discuss relevant related work under the framework described in section 1. We then introduce our proposed PMW and AMW

scheduling policies in section IV. Sections V and VI give the stability results of the proposed policies and evaluate their performances through simulations.

II. System Model

A. Optical Switch Network

As shown in Fig. 1, we consider a set of N top of rack (ToR) switches, labeled by $\{1, 2, \ldots, N\}$, which are interconnected by an optical switched network. Each ToR switch can serve 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 the 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 set $A_{ii}(t) = D_{ii}(t) = L_{ii}(t) = 0$ for all t and 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 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. We also assume that $A_{ij}(t)$ has a finite support, *i.e.* $\exists\ K < \infty$ such that $A_{ij}(t) \leq K$. We call the mean of $A_{ij}(t)$ as the traffic rate $\lambda_{ij} = \mathbb{E}\{A_{ij}(0)\}$, and define the traffic rate matrix as $\boldsymbol{\lambda} = [\lambda_{ij}] \in \mathbb{R}^{N \times N}$.

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. Note that $S_{ii}(t) = 0$ for all t and $i \in \{1, 2, \ldots, N\}$. We also 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. $\sum_i S_{ij}(t) \leq 1, \sum_j S_{ij}(t) \leq 1$. The feasible schedules for the network are determined by the network topology, and we let $\mathcal{F}$ denote the set of all feasible schedules, *i.e.* $\mathbf{S}(t) \in \mathcal{F}$ for all t. Note that if a schedule $\mathbf{S}$ contains N circuit connections, then $\mathbf{S}$ is a permutation matrix. Furthermore, if all such schedule is in the feasible schedule set $\mathcal{F}$, we say the network topology is non-blocking.

B. Stability and Capacity Region

An edge queue Q_{ij} is strongly stable if its queue length $L_{ij}(t)$ satisfies:

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=1}^{t} \mathbb{E}\{\mathbf{L}_{ij}(\tau)\} < \infty$$

and we say the system of queues is stable if Q_{ij} is strongly stable for all $i, j \in \{1, 2, \ldots, N\}, i \neq j$. A scheduling policy is said to stabilize the system if the system is stable under that scheduling policy. With this notion of stability, we define the capacity region $\mathcal{C}$ of the network as the set of all traffic rate matrix such that there exists a scheduling policy which stabilizes the system.

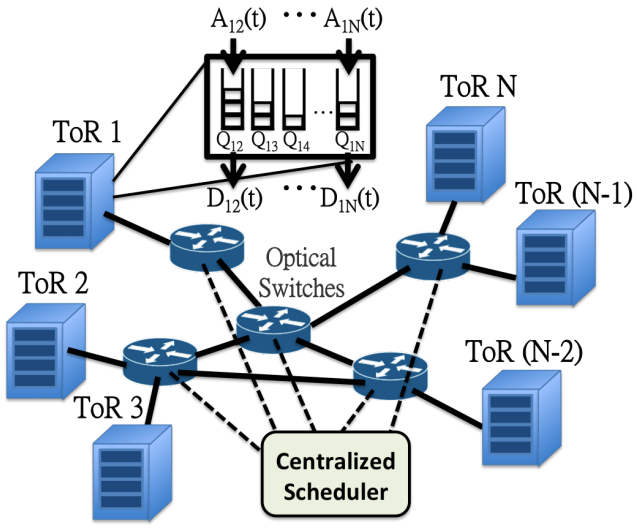

Fig. 1. An Example of the system model

The capacity region is given by the interior of the convex hull of the feasible schedules $\mathcal{F}$ [6], that is

$$\mathcal{C} = \left\{ \sum_{\mathbf{S} \in \mathcal{F}} \alpha_{\mathbf{S}} \mathbf{S} : \sum_{\mathbf{S} \in \mathcal{F}} \alpha_{\mathbf{S}} < 1,\ \alpha_{\mathbf{S}} \geq 0,\ \forall \mathbf{S} \in \mathcal{F} \right\}$$

For any traffic rate matrix $\boldsymbol{\lambda} \in \mathcal{C}$, we say that $\boldsymbol{\lambda}$ is admissible, and define the load of the traffic as $\rho(\boldsymbol{\lambda}) = \max\{r : \boldsymbol{\lambda} \in r\bar{\mathcal{C}},\ 0 < r < 1\}$, where $\bar{\mathcal{C}}$ is the closure of $\mathcal{C}$.

We say that a scheduling policy achieves 100% throughput if it stabilizes the system of edge queues under any admissible traffic.

C. Timing Parameters

One of the main attributes of our work is to identify three distinct time sequences associated with monitoring, computation, and schedule reconfiguration.

Definition 1. Let $\{t_k^M\}_{k=1}^{\infty}$ denote the time instances that the state of edge queues are uploaded to the centralized scheduler. Specifically, the information available at the sheduler is a subset of the edge queue lengths $\{\mathbf{L}(t_k^M)\}_{k=1}^{\infty}$.

Definition 2. Let $\{t_k^C\}_{k=1}^{\infty}$ denote the time instances when a set of new schedules are computed. A scheduler could generate one schedule or multiple schedules, which depends on the scheduling policy used.

Definition 3. 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.*

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

Each of the three processes is associated with a delay as described below.

Definition 4. Let Δ_m be the delay of the monitoring process. This means that the edge queue lengths at time t_k^M, $\mathbf{L}(t_k^M)$, is available at the scheduler after time $t_k^M + \Delta_m$. Therefore, at any time instance t, the edge queue lengths information available at the scheduler is the set $\{\mathbf{L}(t_k^M)\}_{k=1}^n$, where $n = \max\{k : t_k^M + \Delta_m < t\}$.

Definition 5. Let Δ_c be the delay of the computation process of the scheduler generating a set of new schedules. This means that the schedules computed at time t_k^C are available (could be used) after time $t_k^C + \Delta_c$.

Definition 6. Let Δ_r be the reconfiguration delay associated with establishing a new schedule across 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 $D_{ij}(t_k^S + \tau) = 0$.

These timing parameters restrict the scheduling algorithms from using spontaneous edge queue information. For example, the beginning of a schedule at time t, $S(t)$, is actually being reconfigured at time $t - \Delta_r$. The computation of this schedule began at time $t - \Delta_r - \Delta_c$, and the computation is based on information of edge queues at time $t - \Delta_r - \Delta_c - \Delta_m$. In this work, we are primarily interested in the case of $\Delta_c, \Delta_m \approx 0$, $\Delta_r > 0$, even though in Section VI, we will briefly discuss the impact of $\Delta_m > 0$ via simulations.

III. Related Work

An alternative to MaxWeight policy, some prior works in the area of switch scheduling rely on the Birkhoff and von Neumann (BvN) theorem [9] to construct a scheduling policy with 100% throughput. The BvN theorem states that any admissible doubly stochastic matrix can be decomposed as a convex combination of permutation matrices. The BvN scheduling policy [10] assumes the knowledge of the arrival statistics and relies on a BvN decomposition of arrival rate matrix into a set of schedules. The scheduler ensures queue stability by ensuring each schedule in the set is served for an appropriate time interval proportional to the BvN decomposition coefficients.

While BvN scheduler of [10] indeed ensures queue stability for ergodic admissible arrivals, the delay performance can be significantly worse than that of the MaxWeight policy. This performance degradation gets worse with the number of ToR switches: while the delay under the MaxWeight scheduler has an upper bound of $O(N)$, any open loop policy, including the BvN policy of [10], which assigns schedules independent of the queue states is shown to result in delay that has a lower bound of $O(N)$. As a result, in our work we have restricted our attention to the class of closed loop policies.

In [4], a closed loop scheduling policy based on BvN decomposition is proposed. The preposed traffic matrix scheduling (TMS) policy [4] falls in the class of fixed batch scheduling policies proposed in [11] in the context of switching with non-negligible reconfiguration delay, $\Delta_r > 0$. The TMS and fixed batch policies periodically monitor the edge queue (ToR queue) lengths to account for outstanding packets accumulated up to any given scheduling time. The number of outstanding packets in the ToR queues are normalized to generate a fixed number of schedules (according to BvN decomposition) to be used in the pre-specified period till the next monitoring/computation time instance. When necessary, the TMS policy has also been combined with packet switching over an electronic switched network, as suggested in [4].

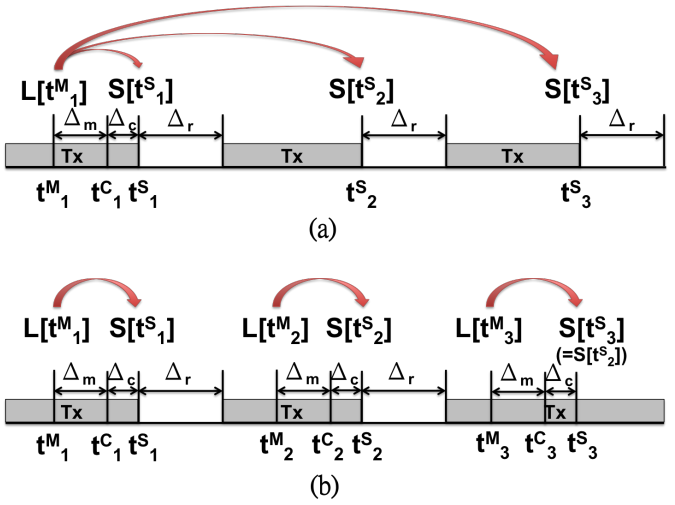

Fig. 2. Timing diagram for different scheduling strategies. (a) Quasi-static monitoring: Series of schedules determined in a single schedule computation, and some schedules could depend on out-dated queue information when being deployed. (b) Active monitoring: Each schedule is computed based on the most up-to-date edge queue information.

The scheduling policies in [4] and [11] both involve "quasi-static monitoring": selecting series of schedules based on a single schedule computation process. When monitoring and computation times are identically coupled, generated schedules may depend on very out-dated information, as shown in Figure 2 (a). We argue that it is always beneficial to employ "active monitoring": schedules must be selected based on frequent and up-to-date queue information, as shown in Figure 2 (b).

In this work, our first contribution is to show that decoupling the monitoring and scheduling rates to allow for active monitoring results in significant improvements. In particular, we show that rather than pre-selecting a set of schedules, and hence a fairly tardy queue monitoring, it is advantageous to allow for "active monitoring" and "frequent computations". In the following sections, we propose policies of active monitoring type, namely the PMW and AMW policies, and analyze their stability and delay performance.

Secondly, to account for the non-negligible reconfiguration delay, hence the loss in the duty cycle, the prior work either rely on the explicit traffic statistics or a conservative upper bound to restrict the rate of schedule reconfigurations. In this paper, a novel adaptive scheduling algorithm is proposed to ensure that schedule reconfigurations occur at optimized time instances.

IV. Scheduling Policies

In this section we introduce three scheduling policies. Under the first two policies, the monitoring time instances are selected such that $t_k^M = kT$, $k \in \mathbb{N}_+$, where T would be the mean schedule duration and is selected appropriately. The difference of the two scheduling policies is in the choice of the computation time instances $\{t_k^C\}$. Under the Traffic Matrix Scheduling (TMS), first introduced in [4] and discussed in subsection IV-A, schedules are computed in a batch such that $t_k^C = kqT = t_{kq}^M = t_{kq}^S$ for a preselected parameter q; while Periodic MaxWeight (PMW) relies on active monitoring to make computation of one schedule at each monitoring time, *i.e.* $t_k^C = t_k^M = t_k^S$. As we will see in sections V and VI, both TMS and PMW policies require that the selection of the parameter T be dependent on arrival statistics. In contrast, the third scheduling policy, the Adaptive MaxWeight, adaptively selects $\{t_l^S\} \subset \{t_k^M\} = \{t_k^C\}$ in a manner to ensure stability and optimized queue lengths.

A. Traffic Matrix Scheduling

In this subsection we briefly describe the traffic matrix scheduling (TMS) policy in [4], which would be a benchmark comparison to our proposed policies. The TMS policy is based on Birkhoff von-Neumann (BvN) Theorem that every doubly stochastic matrix could be decomposed as a convex combination of permutation matrices. The scheduler makes schedule computation every qT slots, where T is the selected mean schedule duration, and q be the number of schedules used between two schedule computation time instances. At the computation time instances $t_k^C = t_{kq}^M$, the scheduler takes the edge queue length matrix $\mathbf{L}(t_{kq}^M)$ and scales it to a doubly stochastic matrix $\mathbf{B}(t_{kq}^M)$ which indicates the relative service requirement in the following qT slots. The scheduler then performs a BvN decomposition [10] on $\mathbf{B}(t_{kq}^M)$:

$$\mathbf{B}(t_{kq}^M) = \sum_{i=1}^{Q} \alpha_i \mathbf{P}_i$$

where each $\mathbf{P}_i$ is a permutation matrix, and is a schedule that would be served for $\alpha_i qT$ slots within the following qT slots. Depending on the demand $\mathbf{B}(t_{kq}^M)$, the number of terms Q in the decomposition may vary ($Q \leq N^2 - 2N + 2$). In practice, we select q largest weighted schedules to avoid excessive schedule changes. Rearrange the order in the decomposition so that $\{\alpha_i\}_{i=1}^q$ are the q largest coefficients. The coefficients are then scaled proportionally to $\tilde{\alpha}_i = \alpha_i / \sum_{i=1}^q \alpha_i$, $i = 1, \ldots, q$.

B. Periodic MaxWeight

The timing sequeces in the PMW policy are selected as $t_k^M = t_k^C = t_k^S = kT$, $k \in \mathbb{N}_+$. The PMW policy computes a schedule at time instance t_k^C based on the edge queue lengths $\mathbf{L}(t_k^M)$. The schedule for time t_k^S is selected as

$$\mathbf{S}\left(t_k^S\right) = \mathbf{S}^*\left(t_k^M\right) = \arg\max_{\mathbf{S}\in\mathcal{F}} \left\langle \mathbf{S}, \mathbf{L}(t_k^M) \right\rangle$$

where $\mathbf{S}^*\left(t_k^M\right) = \langle \mathbf{S}, \mathbf{L}(t_k^M) \rangle = \sum_{i,j=1}^N \mathbf{S}_{ij} \mathbf{L}_{ij}(t_k^M)$ is the weight of a schedule $\mathbf{S}$ at time t. We call $\mathbf{S}^*(t_k^M)$ the MaxWeight schedule at time t_k^M. The scheduler then reconfigures the schedule at time t_k^S, $\mathbf{S}(t_k^S) = \mathbf{S}^*(t_k^M)$.

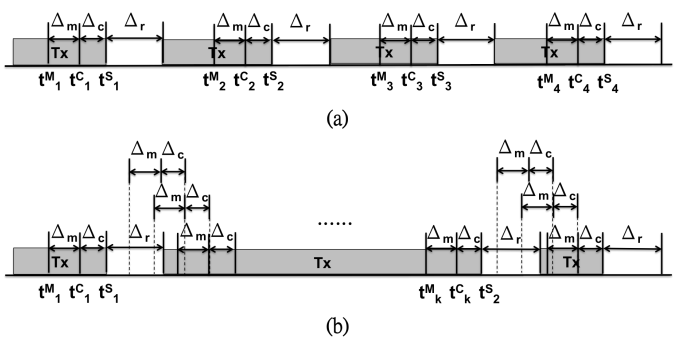

Fig. 3. Timing diagrams for: (a) Periodic scheduling: While each schedule is dependent on the most up-to-date edge queue information, the schedule update is done periodically. (b) Adaptive scheduling: The scheduler frequently monitors the queue information and computes schedule weights. The schedule reconfiguration time instances become aperiodic. Each schedule is computed based on the most up-to-date edge queue lengths.

C. Adaptive MaxWeight

The PMW policy stabilizes any feasible traffic as long as the traffic load is known, as would be shown in the next section. However, in terms of the delay performance, it is not hard to see that the PMW policy might suffer from its periodic schedule update behavior. The schedule may become inefficient before the next reconfiguration time. This would then degrade the throughput of the current schedule.

Based on this observation, we propose the Adaptive MaxWeight (AMW) algorithm, which determines the schedule reconfiguration instances, hence their rate, adaptively (based on instantaneous edge queue information), as shown in Fig. 3. In AMW, we require the scheduler to continuously monitor the edge queues and make a MaxWeight computation (at every time slot t_k^C) based on edge queue lengths $\mathbf{L}(t_k^M)$: In particular, the scheduler computes the weight of the MaxWeight schedule

$$w^*\left(t_k^M\right) = \max_{\mathbf{S}\in\mathcal{F}} \left\langle \mathbf{S}, \mathbf{L}(t_k^M) \right\rangle$$

and the weight of the current schedule

$$w\left(t_k^M\right) = \left\langle \mathbf{S}(t_k^M), \mathbf{L}(t_k^M) \right\rangle.$$

The AMW algorithm keeps track of the difference $\Delta w(t_k^M) = w^*(t_k^M) - w(t_k^M)$ and compares it with a threshold $\sigma(t_k^M) = (1-\gamma)(w^*(t_k^M))^{1-\delta}$, where the **ratio threshold** $\gamma \in (0,1)$ and the **sublinear exponent** $\delta \in [0,1)$ are predetermined system parameters. If $\Delta w(t_k^M)$ is larger than the threshold $\sigma(t_k^M)$, then the scheduler decides to change the schedule to the MaxWeight schedule; otherwise keeps the current schedule. Therefore, the schedule reconfiguration time instance t_l^S is given by

$$t_l^S = \min\left\{ t_k^M : t_k^M > t_{l-1}^S, \Delta w(t_k^M) > \sigma(t_k^M) \right\}$$

and the schedule is reconfigured at time t_l^S as

$$\mathbf{S}(t_l^S) = \arg\max_{\mathbf{S}\in\mathcal{F}} \left\langle \mathbf{S}, \mathbf{L}(t_k^M) \right\rangle$$

where t_k^M is the monitoring time instance that corresponds to the schedule reconfiguration time t_l^S.

Notice under the Adaptive MaxWeight the times to perform schedule reconfiguration are not limited to the multiples of T,

which had to be chosen with some knowledge of Δ_r and traffic load. Instead the schedule reconfigurations can occur sooner or later depending on the current state of the system, which in turn is a result of prior effective switching rate. The proposed AMW policy relies on the fact that we have decoupled the restrictions on the decision process (monitoring and computation) from the restriction on the action process (reconfiguration). We provide theoretical guarantees on throughput optimality of the AMW policy in section V. In section VI we investigate the delay performance of the proposed scheduling policies empirically.

V. Stability Analysis

In this section we analyze the stability of the proposed scheduling policies, the PMW and AMW policies. The reconfiguration delay is assumed to be nonzero, *i.e.* $\Delta_r > 0$, while the monitoring and computation delay are assumed to be negligible, $\Delta_m = \Delta_c \approx 0$. We consider the irreducible discrete time Markov chain (DTMC) process describing the evolution of edge queue occupancies $\mathbf{L}(t)$ and prove that it satisfies the Foster Lyapunov Theorem (see Appendix).

A. Periodic MaxWeight

Due to the overhead that incurred by the reconfiguration time Δ_r, the duty cycle of the PMW algorithm with schedule reconfigure period T can be determined as $1 - \frac{\Delta_r}{T}$. Theorem 1 establishes that given an admissible traffic load ρ, the PMW algorithm guarantees stability as long as the schedule reconfigure period T satisfies $1 - \frac{\Delta_r}{T} > \rho$ (or equivalently, $T > \frac{\Delta_r}{1-\rho}$).

Theorem 1. *Given an admissible traffic $\boldsymbol{\lambda}$ with load $\rho(\boldsymbol{\lambda}) < 1$ and the reconfiguration delay Δ_r. If the schedule reconfigure period of the PMW policy satisfies $T > \frac{\Delta_r}{1-\rho}$, then the PMW policy strongly stabilizes the edge queues.*

The proof is given in appendix A.

B. Adaptive MaxWeight

In the AMW algorithm, schedule changes do not necessarily occur at the beginning of each interval $[t_k, t_{k+1}]$. Instead, as the queue occupancies grow, the time between two schedule reconfigurations may in general increases. In other words, a drop in duty cycle would delay the packets and create a large queue; this increase in queue size is then fed back to increase the threshold $\sigma(t)$ and hence reduce the rate of reconfigurations. We then utilize this observation to establish a lower bound for the time between two schedule reconfigurations and derive the stability. Additionally, the AMW policy has the advantage that during low load periods, it reconfigures the MaxWeight scheduling at a higher reconfiguration rate, resulting in a better delay performance.

Theorem 2. *For any admissible traffic load ρ, if the ratio threshold $\gamma \in (0, 1)$ and the sublinear exponent $\delta \in (0, 1)$, then the AMW policy strongly stabilizes the edge queues.*

The proof is given in appendix B.

Finally, notice that we omitted Δ_m and Δ_c in the proof of stability. Referring to the proof of stability under MWM with a fixed delay in [12], we claim that when Δ_m and Δ_c are fixed values, the stability results proved in this section can be trivially generalized.

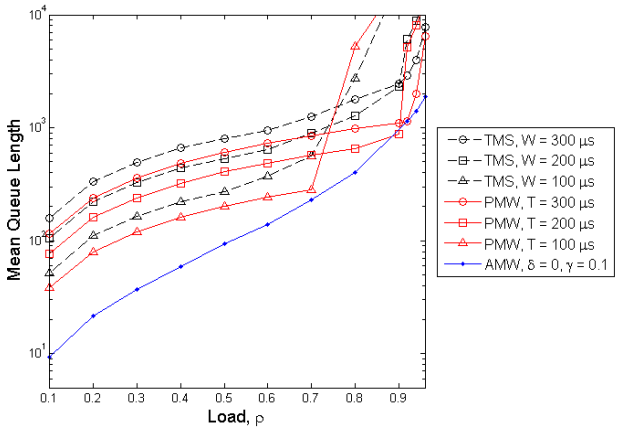

Fig. 4. Mean queue length versus traffic load ρ under the **uniform traffic**. The TMS policy reconfigures the schedule $q = 10$ times within qT time duration. The scheduling rate under either the TMS or PMW is equal to $1/T$, while under AMW is adapted to the traffic load intensity.

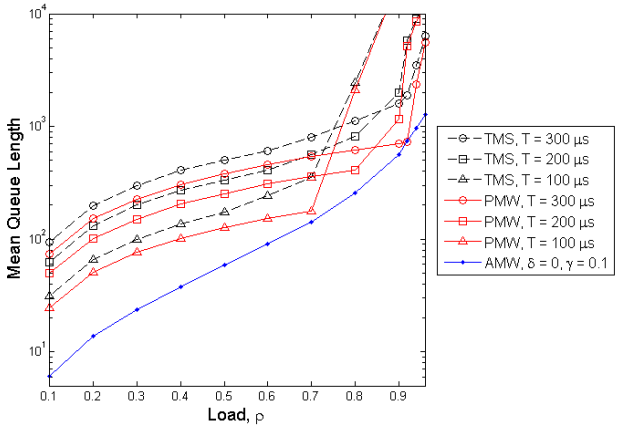

Fig. 5. Mean queue length versus traffic load ρ under the **nonuniform traffic**. The TMS policy reconfigures the schedule $q = 10$ times within qT time duration.

VI. Performance Evaluation

In this section we present simulation results for the proposed PMW and AMW policies, and compare them to the benchmark scheduling policy TMS.

The experiments are conducted with the simulator built for the REACToR switch in [13]. 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 design in [13] 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 $\mathcal{F}$ is in fact the set of $N \times N$ permutation matrices. Each link has data bandwidth of $B = 100$ 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

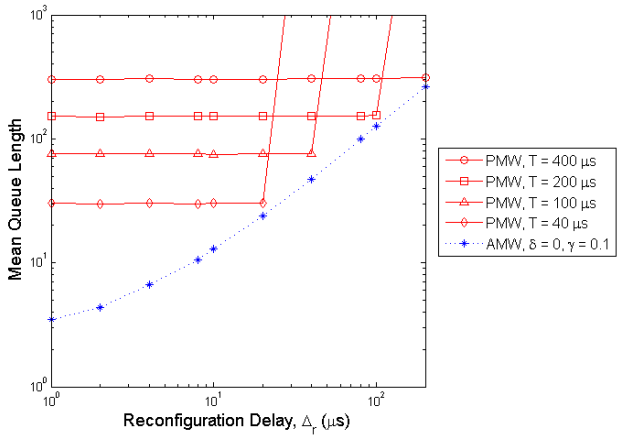

Fig. 6. Mean queue length versus the reconfiguration delay Δ_r under the nonuniform traffic. The traffic load is fixed as $\rho = 0.3$. We assume negligible monitoring and computation delay, $\Delta_m = \Delta_c \approx 0$.

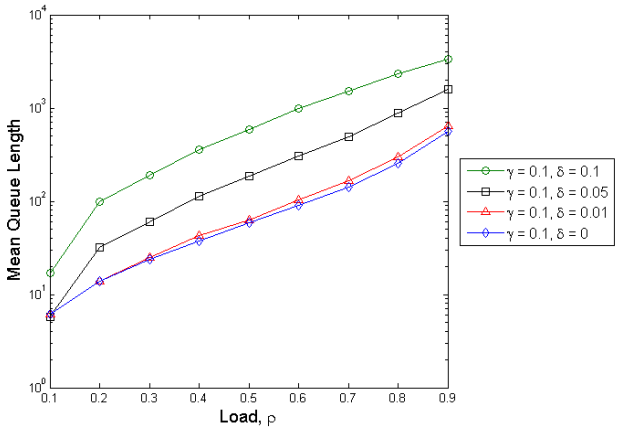

Fig. 7. Mean queue length versus traffic load for the AMW under different sublinear exponent δ.

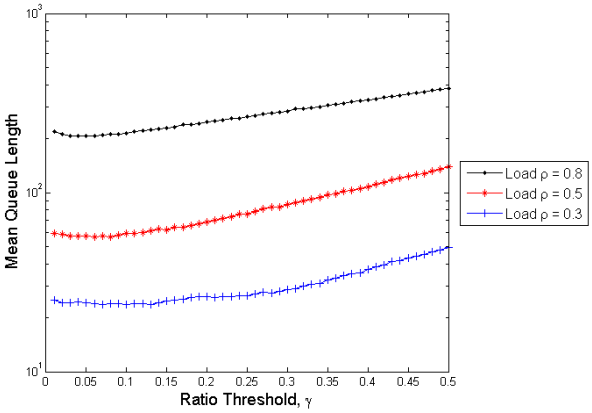

Fig. 8. Mean queue length of AMW alogirthm versus the threshold value γ. The performance is robust under light to medium traffic loads. Small values of γ cause large delays as the load increases.

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 $\boldsymbol{\lambda}$ used in this section are classified to the following types:

1) Uniform: $\lambda_{ij} = \rho/N,\ \forall 1 \le i, j \le N$.
2) Nonuniform: $\lambda_{ij} = \frac{\rho}{M} \sum_{m=1}^{M} \mathbf{P}_{ij}^m$ where $\mathbf{P}^m, m = 1, \ldots, M \in \mathcal{P}$ are permutation matrices picked at random. The number M determines the skewness of the load matrix. We set $M = 100$ here.

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

In Figs 4 and 5, we show performance comparison of the three scheduling algorithms 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 PMW perform comparably with the PMW slightly outperforming the TMS under the same schedule reconfiguration rate $1/T$. We note that under both the TMS and PMW policies, the traffic loads they could stabilize are determined by the reconfiguration rate $1/T$. In general, a smaller T value gives better delay performance at a fixed load, but choosing a smaller T value also decreases the maximum load that the TMS or PMW policy could stabilize. On the other hand, the AMW policy always outperforms the PMW and TMS.

We now consider the effect of the reconfiguration delay Δ_r to the performance. In Fig. 6, we show the performance of the PMW and AMW under various Δ_r, while the traffic load is fixed as $\rho = 0.3$. The performance We can see that the performance of the AMW outperforms the PMW under each Δ_r value, regardless of the parameter selection of the PMW policy. Although there exists an optimal schedule reconfiguration period T of the PMW policy that achieves comparable performance to the AMW policy at each Δ_r, the choice of the optimal T is dependent on the traffic load ρ. We can see that the performance of the AMW actually traces the optimal performance of the PMW. This observation suggests that the adaptive strategy of the AMW in fact allows it to capture the optimal schedule reconfiguration rate based solely on the queue lengths information and no prior knowledge of the arrival statistics is required.

Finally we consider the effect of the parameter selection for the AMW policy. Fig. 7 shows the performance for different values of sublinear exponent δ. Note that the mean queue length becomes shorter when δ is smaller. Although the stability for the case $\delta = 0$ is not covered in the previous section, we see from the simulation that it is stable for admissible traffic and has the best performance. We then consider the selection of the ratio threshold γ. Fig. 8 presents the mean queue length varying with the ratio threshold γ under different traffic loads $\rho = \{0.3, 0.5, 0.8\}$. We may observe that the performance is fairly smooth under wide range of γ value. The optimal value of γ varies slightly with the traffic load ρ. However, in the region $\gamma \in [0.05, 0.1]$, the delay performance is generally well across different traffic loads.

VII. Conclusion

This paper considered the end-to-end scheduling problem in all-optical data center networks. The entire network can be viewed as a generalized crossbar interconnect with nonzero schedule reconfiguration delay. We first proposed a decoupling of three fundamental time series: namely, the monitoring time instances $\{t_k^M\}$, the computation time instances $\{t_k^C\}$, and the schedule reconfiguration time instances $\{t_k^S\}$. This decoupling of the monitoring rate and the schedule reconfiguration rate results in a performance gain associated with "active monitoring" and "frequent computations". Utilizing an active monitoring paradigm, the Periodic MaxWeight (PMW) and the Adaptive MaxWeight (AMW) policies were proposed. The proposed policies are shown to achieve queue stability via theoretical analysis and to outperform the benchmark scheduling policy TMS through simulations.

The second contribution of this work is to show the benefit of the adaptive scheduling in all-optical data center networks with reconfiguration delay. The proposed adaptive policy AMW illustrates that utilizing the adaptive scheme, a scheduling policy could achieve the full stability region without prior knowledge of the traffic statistics. In contrast, the periodic scheduling policy PMW requires the prior knowledge of the traffic load in order to guarantee stability of the network. The stability guarantees are established both analytically and empirically through simulations under i.i.d. arrival traffic.

The proposed scheduling policies in this work considers zero in-network buffer due to the inherently bufferless nature of the optical circuits. It is interesting to note that the notion of edge-buffering and end-to-end scheduling has also been explored in the regime of electronic packet-switched data center network [14] recently, in an effort to reduce buffering and congestion within the network. This suggests that our proposed scheduling policies can also be utilized in the context of electronic packet switches (with in-network buffering) in order to further reduce delay and improve performance.

In this work we consider primarily the case of negligible monitoring delay and computation delay, *i.e.* $\Delta_m, \Delta_c \approx 0$. In practice, however, we usually have $\Delta_c, \Delta_m > 0$ and especially Δ_m to grow with respect to the scale of the network or the monitoring system used. Although Δ_m do not affect the system stability, the value of Δ_m does affect the delay performance of scheduling policies. As seen in Fig. 9, in the small Δ_m regime, the AMW policy achieves substantially better performance over the comparing schedulig policies. In contrast, as Δ_m increases, the performance of the AMW policy sees a siginificant degradation. This observation motivates two directions for future research: 1) the development of low-delay ToR monitoring system and 2) improvements to the AMW policy in order to increase the robustness with respect to the monitoring delay Δ_m.

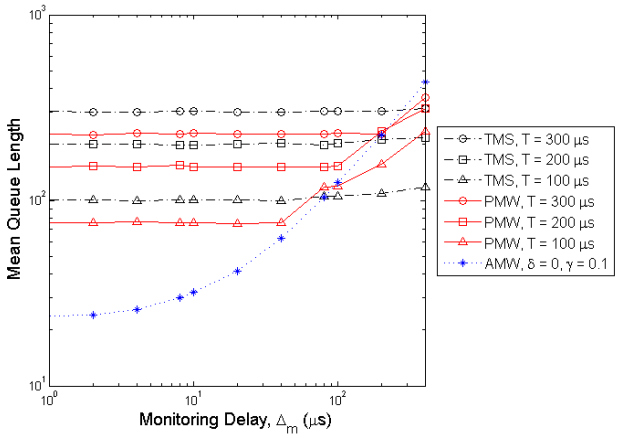

Fig. 9. Mean queue length versus monitoring delay Δ_m under nonuniform traffic. The traffic load is fixed as $\rho = 0.3$. The edge queue state is monitored/updated every microsecond, $t_{k+1}^M - t_k^M = 1\mu$s for all k.

Acknowledgment

This work has been partially supported by L3 Communications and NSF Center for Integrated Access Networks (NSF Grant EEC-0812072).

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity, data center network architecture," in *Proc. ACM SIGCOMM*, 2008.

[2] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers," in *Proc. ACM SIGCOMM*, Aug. 2010.

[3] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," in *Proc. ACM SIGCOMM*, Aug. 2010.

[4] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, (New York, NY, USA), pp. 447–458, ACM, 2013.

[5] "Nistica Wavelength Selective Switch Product Data Sheet." http://www.nistica.com/products.html.

[6] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *Automatic Control, IEEE Transactions on*, vol. 37, pp. 1936–1948, Dec 1992.

[7] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, vol. 1, pp. 296–302 vol.1, Mar 1996.

[8] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 556–564 vol.2, 2000.

[9] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser. A5*, no. 147-150, 1946.

[10] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, "Birkhoff-von neumann input buffered crossbar switches," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1614–1623 vol.3, Mar 2000.

[11] 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.), pp. 65–79, Springer London, 2007.

[12] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 792–799, Mar 1998.

[13] H. Liu, F. Lu, A. Forencich, R. Kapoor, M. Tewari, G. M. Voelker, G. Papen, A. C. Snoeren, and G. Porter, "Circuit switching under the

radar with reactor," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, (Berkeley, CA, USA), pp. 1–15, USENIX Association, 2014.

[14] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A Centralized Zero-Queue Datacenter Network," in *ACM SIGCOMM 2014*, (Chicago, IL), August 2014.

[15] E. Leonardi, M. Mellia, F. Neri, and M. Ajmone Marsan, "Bounds on average delays and queue size averages and variances in input-queued cell-based switches," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 1095–1103 vol.2, 2001.

[16] Y. Ganjali, A. Keshavarzian, and D. Shah, "Input queued switches: cell switching vs. packet switching," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, pp. 1651–1658 vol.3, March 2003.

APPENDIX

Consider the dynamics of the edge queue lengths $\mathbf{L}(t)$ at the stopping times t_k as

$$\mathbf{L}(t_{k+1}) = \mathbf{L}(t_k) + \sum_{t=1}^{T} \Big[\mathbf{A}(t_k+t) - \mathbf{D}(t_k+t)\Big] \quad (1)$$

In the proofs of this appendix, we consider the quadratic Lyapunov function for the edge queue lengths as $V(\mathbf{L}) = \langle \mathbf{L}, \mathbf{L}\rangle = \sum_{i=1}^{N}\sum_{j=1}^{N} \mathbf{L}_{ij}^2$, and show that $\mathbf{L}(t)$ satisfies the following Foster-Lyapunov Theorem:

Fact 1 (Foster-Lyapunov [15])**.** *Given a system of edge queues $Q_{ij}, 1 \le i,j \le N$, with queue occupancies $\mathbf{L}(t) = [L_{ij}(t)]$. Let $\{t_k\}$ be a sequence of stopping times. Let there exist positive real numbers $\epsilon > 0$ and $B > 0$, and a lower bounded, real-valued Lyapunov function $V(\mathbf{L})$ such that*

$$(1)\mathbb{E}\left[V(\mathbf{L}(t_{k+1}))|\mathbf{L}(t_k)\right] < \infty, \quad \forall \mathbf{L}(t_k)$$
$$(2)\mathbb{E}\left[V\left(\mathbf{L}(t_{k+1})\right) - V(\mathbf{L}(t_k))|\mathbf{L}(t_k)\right] < -\epsilon||\mathbf{L}(t_k)||, \quad \forall ||\mathbf{L}(t_k)|| > B$$

where $||\mathbf{L}|| = \sqrt{\langle \mathbf{L}, \mathbf{L}\rangle}$ is the 2-norm of queue occupancies. Then the system of edge queues is strongly stable.

A. Proof of Theorem 1

Proof: We select T such that $T > \frac{\Delta_r}{1-\rho}$, and define the sequence of stopping times as $t_k = kT$. The expected drift of the Lyapunov function is then given by

$$\begin{aligned}&\mathbb{E}\left[V\left(\mathbf{L}(t_{k+1})\right) - V(\mathbf{L}(t_k))|\mathbf{L}(t_k)\right]\\ =&\mathbb{E}\left[\Big\langle \mathbf{L}(t_{k+1}), \mathbf{L}(t_{k+1})\Big\rangle - \Big\langle \mathbf{L}(t_k), \mathbf{L}(t_k)\Big\rangle \Big| \mathbf{L}(t_k)\right]\\ =&\mathbb{E}\left[\Big\langle 2\mathbf{L}(t_k) + \Delta\mathbf{L_k}, \Delta\mathbf{L_k}\Big\rangle \Big| L(t_k)\right] \end{aligned} \quad (2)$$

where $\Delta\mathbf{L_k} = \sum_{t=1}^{T}\left(\mathbf{A}(t_k+t) - \mathbf{D}(t_k+t)\right)$. The assumption of finite support of the arrival process then implies the finiteness of $\Delta\mathbf{L_k}$, hence there exists a bound $C < \infty$ such that $\Big\langle \Delta\mathbf{L_k}, \Delta\mathbf{L_k}\Big\rangle \le C$.

Since the first Δ_r time slots in the $[t_k, t_{k+1}]$ interval is the reconfiguration time, $\mathbf{D}(t_k+t) = 0$ for $t \in [0, \Delta_r]$, we have

$$\begin{aligned}&\mathbb{E}\left[\Big\langle \mathbf{L}(t_k), \Delta\mathbf{L_k}\Big\rangle \Big| \mathbf{L}(t_k)\right]\\ =&\sum_{t=1}^{T}\Big\langle \mathbf{L}(t_k), \boldsymbol{\lambda}\Big\rangle - \sum_{t=\Delta_r+1}^{T}\mathbb{E}\left[\Big\langle \mathbf{L}(t_k), \mathbf{D}(t_k+t)\Big\rangle \Big| \mathbf{L}(t_k)\right]\\ =&\sum_{t=\Delta_r+1}^{T}\Big\langle \mathbf{L}(t_k), \frac{T}{T-\Delta_r}\boldsymbol{\lambda} - \mathbf{S}^*(t_k)\Big\rangle\\ &+\sum_{t=\Delta_r+1}^{T}\mathbb{E}\left[\Big\langle \mathbf{L}(t_k), \mathbf{S}^*(t_k) - \mathbf{D}(t_k+t)\Big\rangle \Big| \mathbf{L}(t_k)\right] \end{aligned} \quad (3)$$

Let $\boldsymbol{\lambda}_r = \frac{T}{T-\Delta_r}\boldsymbol{\lambda}$, then since the traffic load is ρ and $\frac{T}{T-\Delta_r} = \frac{1}{1-\frac{\Delta_r}{T}} < \frac{1}{\rho}$, we have $\boldsymbol{\lambda}_r < \frac{1}{\rho}\boldsymbol{\lambda}$ and thus $\boldsymbol{\lambda}_r \in \mathcal{C}$. We may then write $\boldsymbol{\lambda}_r = \sum_l \alpha_l \mathbf{S}_l$, where $\sum_l \alpha_l < 1$ and $\mathbf{S}_l \in \mathcal{F}$ for each l. Let $\delta = 1 - \sum_l \alpha_l$ then we have

$$\begin{aligned}\Big\langle \mathbf{L}(t_k), \boldsymbol{\lambda}_r - \mathbf{S}^*(t_k)\Big\rangle &\le (\sum_l \alpha_l - 1)w^*(t_k)\\ &= -\delta w^*(t_k)\end{aligned} \quad (4)$$

Note that $\mathbf{S}^*(t_k)$ is the schedule used during the period $[t_k, t_{k+1}]$. For $t \in [\Delta_r, T]$, we have $\mathbf{D}_{ij}(t_k+t) \ne \mathbf{S}^*_{ij}(t_k+t)$ only if $\mathbf{L}_{ij}(t_k+t-1) = 0$, which then implies $\mathbf{L}_{ij}(t_k) \le T$ since at most one packet could depart from an edge queue at each time slot. We then have the following bound:

$$\begin{aligned}&\mathbb{E}\left[\sum_{t=\Delta_r+1}^{T}\Big\langle \mathbf{L}(t_k), \mathbf{S}^*(t_k) - \mathbf{D}(t_k+t)\Big\rangle \Big| \mathbf{L}(t_k)\right]\\ \le &\sum_{t=\Delta_r+1}^{T}\sum_{i,j=1}^{N} T\ \mathbf{S}^*_{ij}(t_k) \le (T-\Delta_r)NT\end{aligned} \quad (5)$$

Combining eqs. (2) - (4), we have

$$\begin{aligned}&\mathbb{E}\left[V\left(\mathbf{L}(t_{k+1})\right) - V(\mathbf{L}(t_k))|\mathbf{L}(t_k)\right]\\ \le &-2\delta(T-\Delta_r)w^*(t_k) + 2(T-\Delta_r)NT + C\end{aligned}$$

Then since $w^*(t_k) \ge \frac{1}{N}||\mathbf{L}(t_k)||$, Fact 1 establishes the strong stability.

■

B. Proof of Theorem 2

Before we proceed to the proof of Theorem 2, we use the following definition and fact for the outdated schedules [16] to establish the relation between the edge queue lengths and the time between two schedule reconfigurations.

Definition 7. At time instance t, the current schedule $\mathbf{S}(t)$ is a p-outdated schedule if it is a maximum weight schedule at time $t-p$, i.e. $\mathbf{S}(t) = \mathbf{S}^*(t-p)$.

Fact 2 (Lemma 1 of [16])**.** *At any time instance t, suppose the current schedule $\mathbf{S}(t)$ is a p-outdated schedule, then the difference between the weight of $\mathbf{S}(t)$ and the maximum weight at time t is at most $(K+1)pN$, i.e.*

$$w(t) \ge w^*(t) - (K+1)pN$$

where K is the bounding constant of the arrival process $\mathbf{A}(t)$.

Given the sequence of stopping times $\{t_k\}$ with $t_k = kT$, Fact 2 would imply that when edge queue lengths are large enough, the time between two schedule reconfigurations would be larger than T. Specifically, we show in the following that if a schedule reconfiguration occurs at time t_c, and $w^*(t_c) \geq \left(\frac{1}{1-\gamma}(K+1)TN\right)^{\frac{1}{1-\delta}} + TN$, then no schedule reconfigurations could occur during the time interval $[t_c, t_c+T]$:

Since at most N queues are served each time slot, then $w^*(t_c+T) \geq \left(\frac{1}{1-\gamma}(K+1)TN\right)^{\frac{1}{1-\delta}}$, and by Fact 2

$$\begin{aligned} & w(t_c+T) \geq w^*(t_c+T) - (K+1)TN \\ \Rightarrow\ & \Delta w(t_c+T) \leq (K+1)TN \leq (1-\gamma)\left(w^*(t_c+T)\right)^{1-\delta} \end{aligned}$$

then no reconfiguration would occur within $[t_c, t_c+T]$. Hence the condition of large edge queue lengths stated above restricts the reconfiguration frequency. We now start the proof of Theorem 2 with this result.

Proof of Theorem 2:

Similar to the proof in appendix A, we consider stopping times $t_k = kT$, where $T > \frac{\Delta_r}{1-\rho}$. The dynamics of the edge queue lengths at the stopping times is also given by 1.

Consider an interval $[t_k, t_{k+1}]$ where $\mathbf{L}(t_k)$ satisfies $\max_{i,j} \mathbf{L}_{ij}(t_k) \geq \frac{1}{1-\gamma}(K+1)TN$. The schedule reconfiguration occurs at most once in the interval $[t_k, t_{k+1}]$ as discussed above. Assuming that the schedule reconfigures at time $t_k + d$ (consider $d = T$ if no schedule change occurs in the interval), then the schedules used in the interval can be expressed as

$$\mathbf{S}(t_k+t) = \begin{cases} \mathbf{S}(t_k), & \text{if } 1 \leq t \leq d \\ 0, & \text{if } d < t \leq d + \Delta_r \\ \mathbf{S}^*(t_k+d), & \text{if } d+\Delta_r < t \leq T \end{cases}$$

we then have

$$\begin{aligned} & \mathbb{E}\left[\left\langle \mathbf{L}(t_k), \Delta\mathbf{L_k}\right\rangle \middle| \mathbf{L}(t_k)\right] \\ =& \sum_{t=1}^{T} \mathbb{E}\left[\left\langle \mathbf{L}(t_k), \boldsymbol{\lambda} - \mathbf{D}(t_k+t)\right\rangle \middle| \mathbf{L}(t_k)\right] \\ \leq & \sum_{\substack{t=1 \\ t\notin[s+1, s+\Delta_r]}}^{T} \Big\{ \left\langle \mathbf{L}(t_k), \boldsymbol{\lambda}_r - \mathbf{S}(t_k+t)\right\rangle \\ & + \mathbb{E}\left[\left\langle \mathbf{L}(t_k), \mathbf{S}(t_k+t) - \mathbf{D}(t_k+t)\right\rangle \middle| \mathbf{L}(t_k)\right] \Big\} \end{aligned}$$

where $\boldsymbol{\lambda}_r = \frac{T}{T-\Delta_r}\boldsymbol{\lambda}$. The second term is bounded similarly as in (5), hence we have a constant $C' < \infty$ such that $\sum_{t=1}^{T} \mathbb{E}\left[\left\langle \mathbf{L}(t_k), \mathbf{S}(t_k+t) - \mathbf{D}(t_k+t)\right\rangle \middle| \mathbf{L}(t_k)\right] \leq C'$. We now give bounds for the first term.

For $1 \leq t \leq d$:

$$\begin{aligned} & \left\langle \mathbf{L}(t_k), \boldsymbol{\lambda}_r - \mathbf{S}(t_k+t)\right\rangle \\ =& \left\langle \mathbf{L}(t_k), \boldsymbol{\lambda}_r - \mathbf{S}^*(t_k)\right\rangle + \left\langle \mathbf{L}(t_k), \mathbf{S}^*(t_k) - \mathbf{S}(t_k)\right\rangle \\ \leq & -\beta w^*(t_k) + (1-\gamma)\left(w^*(t_k)\right)^{1-\delta} \end{aligned} \tag{6}$$

since $\left\langle \mathbf{L}(t_k), \mathbf{S}^*(t_k) - \mathbf{S}(t_k)\right\rangle = w^*(t_k) - w(t_k) \leq (1-\gamma)\left(w^*(t_k)\right)^{1-\delta}$.

For $d + \Delta_r < t \leq T$:

$$\begin{aligned} & \left\langle \mathbf{L}(t_k), \boldsymbol{\lambda}_r - \mathbf{S}(t_k+t)\right\rangle \\ =& \left\langle \mathbf{L}(t_k), \boldsymbol{\lambda}_r - \mathbf{S}^*(t_k)\right\rangle + \left\langle \mathbf{L}(t_k), \mathbf{S}^*(t_k) - \mathbf{S}(t_k+d)\right\rangle \\ \leq & -\beta w^*(t_k) + d(K+1)N \end{aligned} \tag{7}$$

The bound $\left\langle \mathbf{L}(t_k), \mathbf{S}^*(t_k) - \mathbf{S}(t_k+d)\right\rangle \leq d(K+1)N$ follows the similar idea in Fact 2 and is omitted here.

Then with (6), (7), and note that $0 \leq d \leq T$, we have

$$\begin{aligned} & \mathbb{E}\left[\left\langle \mathbf{L}(t_k), \Delta\mathbf{L_k}\right\rangle \middle| \mathbf{L}(t_k)\right] \\ \leq & -\beta(T-\Delta_r)w^*(t_k) + d(1-\gamma)\left(w^*(t_k)\right)^{1-\delta} \\ & + (T-d-\Delta_r)d(K+1)N + C' \\ \leq & -w^*(t_k)\left[\beta(T-\Delta_r) - T(1-\gamma)(w^*(t_k))^{-\delta}\right] \\ & + T(T-\Delta_r)(K+1)N + C' \end{aligned} \tag{8}$$

Combining eqs. (2) and (8), we have

$$\begin{aligned} & \mathbb{E}\left[V\left(\mathbf{L}(t_{k+1})\right) - V(\mathbf{L}(t_k)) | \mathbf{L}(t_k)\right] \\ \leq & -2w^*(t_k)\left[\beta(T-\Delta_r) - T(1-\gamma)(w^*(t_k))^{-\delta}\right] \\ & + 2T(T-\Delta_r)(K+1)N + 2C' + C \end{aligned} \tag{9}$$

Since $w^*(t_k) \leq \frac{1}{N}||\mathbf{L}(t_k)||$ and $\delta > 0$, we have that $(w^*(t_k))^{-\delta}$ is small when the norm of the queue occupancies $||\mathbf{L}(t_k)||$ is large. The stability is then constructed by Fact 1. ■