Energy-aware Scheduling Algorithms for Network Stability

Matthew Andrews
Bell Labs, Murray Hill, NJ
andrews@research.bell-labs.com

Spyridon Antonakopoulos
Bell Labs, Murray Hill, NJ
spyros@research.bell-labs.com

Lisa Zhang
Bell Labs, Murray Hill, NJ
ylz@research.bell-labs.com

***Abstract*—A key problem in the control of packet-switched data networks is to schedule the data so that the queue sizes remain bounded over time. Scheduling algorithms have been developed in a number of different models that ensure network stability as long as no queue is inherently overloaded. However, this literature typically assumes that each server runs at a fixed maximum speed. Although this is optimal for clearing queue backlogs as fast as possible, it may be suboptimal in terms of energy consumption. Indeed, a lightly loaded server could operate at a lower rate, at least temporarily, to save energy.**

Within an energy-aware framework, a natural question arises: "What is the minimum energy that is required to keep the network stable?" In this paper, we demonstrate the following results towards answering that question.

Starting with the simplest case of a single server in isolation, we consider three types of rate adaptation policies: a heuristic policy, which sets server speed depending on queue size only, and two more complex ones that exhibit a tradeoff between queue size and energy usage. We also present a lower bound on the best such tradeoff that can possibly be achieved.

Next, we study a general network environment and investigate two scenarios. In a temporary sessions scenario, where connection paths can rapidly change over time, we propose a combination of the above rate adaptation policies with the standard Farthest-to-Go scheduling algorithm. This approach provides stability in the network setting, while using an amount of energy that is within a bounded factor of the optimum. In a permanent sessions scenario, where connection paths are fixed, we examine an analogue of the well-known Weighted Fair Queueing scheduling policy and show how delay bounds are affected under rate adaptation.

I. Introduction

In this paper we revisit a number of classical network scheduling problems and examine how they are affected when we introduce energy awareness. Most such scheduling problems have been formulated under the assumption that the processing rate of a server is fixed. However, modern servers often allow for dynamic adjustment of their processing rate, because operating at a lower rate typically requires less power. This has inspired a growing body of research on how to best take advantage of this so-called *rate adaptation* capability in the interest of maximizing energy savings.

Herein, our main focus is whether it is possible to maintain stability, or equivalently bounded queues, in a network of servers in an energy-aware manner. In particular, we investigate the tradeoff between energy consumption and performance measures such as delay and queue size.

We remark that there exists an extensive literature proposing various scheduling algorithms that keep the network stable, as long as the servers are always operating at maximum rate. In order to study the problem we posed earlier, though, our task is to determine how to adapt these algorithms so as to minimize energy while preserving stability. The related objective of guaranteeing bounded end-to-end packet delay is also considered.

A. Modeling

First of all, let us describe our modeling of the problem. We consider a network of multiple servers, which process and then forward incoming data traffic. Every server's processing rate may be adjusted independently, taking any value in the interval $[R_{\min}, R_{\max}]$[1], where $R_{\min}$ and $R_{\max}$ are given parameters such that $0 < R_{\min} < R_{\max}$. Furthermore, the power consumed by the server e while operating at rate r_e is given by a so-called *energy function* $f(r_e)$. We adopt the common assumption that $f(s) = s^\alpha$ for some parameter $\alpha > 1$, which is based on properties of CMOS circuits [9], [16].

For every server, a *rate-adaptive scheduling algorithm* makes two decisions at any given time, namely setting the processing rate and determining which data in the server queue to forward. We concentrate on *work-conserving* algorithms only; in other words, the algorithm cannot order a server not to forward any traffic (for whatever reason) at a time when the latter's queue is non-empty. Moreover, with regard to network traffic we consider two standard models, defined below.

1) Temporary Sessions Model: This model is also known as the Adversarial Queuing Model (AQM). It aims to capture situations in which the set of sessions (or connections) carried by the network is highly volatile. The model stipulates that packets are injected into the network by some adversarial process $\mathcal{A}$, henceforth referred to simply as the adversary. Additionally, $\mathcal{A}$ specifies the path along which each packet must be routed at the time of its injection, and the hop count of every such path is bounded by a parameter d. Let $A_e(t, t')$ be the total size of packets injected into the network during the time interval $[t, t')$ that include server e on their paths. In order for each server not to be inherently overloaded, the packet injection by $\mathcal{A}$ is restricted to be $(\sigma, 1-\varepsilon)$-*admissible*, which means that for all e and all intervals $[t, t')$,

$$A_e(t, t') \leq \sigma + (1-\varepsilon) R_{\max}(t' - t) \ ,$$

[1]In practice, only a discrete set of rates would likely be available, due to hardware constraints. Nevertheless, we extend it to a continuous range for clarity of exposition. This entails no loss of generality for our results.

where $\sigma \geq 0$ is the burst size and $\varepsilon > 0$ reflects the load factor. We then say that $\mathcal{A}$ is a *bounded adversary of rate* $(\sigma, 1-\varepsilon)$.

2) Permanent Sessions Model: This model is sometimes referred to as the connection-oriented model, and it reflects a setting where all data is transported along pre-defined connections. Each connection i is specified by a path P_i, a burst size σ_i, and an injection rate ρ_i. Let $H_i(t,t')$ be the total injection into connection i during the time interval $[t,t')$. Again, to ensure servers are not inherently overloaded, the injection is restricted to be admissible in the following sense.

$$\begin{aligned} H_i(t,t') &\leq \sigma_i + \rho_i(t'-t) && \forall i, \forall [t,t') \\ \sum_{i:e\in P_i} \rho_i &\leq (1-\varepsilon)R_{\max} && \forall e \end{aligned} \tag{1}$$

Remark. It is easy to see that any traffic admissible to the permanent sessions model is also admissible to the temporary sessions model. However, the converse statement is not true. Indeed, consider two paths P_1 and P_2 that share a server e. Let us partition time into intervals of arbitrarily large lengths. The temporary sessions model allows injections that alternate between only injecting along P_1 during odd-numbered intervals and only injecting along P_2 during even-numbered intervals, at rate $0.9R_{\max}$ in both cases. This cannot happen in the permanent sessions model, because the connection rates must be set at $\rho_1 = \rho_2 = 0.9R_{\max}$. As a result, $\sum_{i:e\in P_i} \rho_i > R_{\max}$, which would violate (1). (Note that since the interval lengths are arbitrarily large, admissibility in the permanent sessions model cannot be achieved by setting a large burst size for each connection.) Therefore, strictly more traffic injections are admissible in the temporary sessions model.

We say that the network is *stable* if the aggregate queue size remains bounded over time. Our goal in this paper is to maintain stability while also tailoring server rates to traffic loads in such a way that energy usage is minimized. For example, if the long-term traffic through server e satisfies $A_e(t,t') \ll R_{\max}(t'-t)$ for $t' \gg t$, then the latter can operate at a rate much smaller than $R_{\max}$ without jeopardizing stability. However, there is no way to know exactly how much traffic will arrive at e in the future, and thus it is not clear *a priori* how to set its rate for optimal energy efficiency. This constitutes a scheduling problem which, to the best of our knowledge, has not been addressed in conjunction with network stability before.

B. Previous Work

1) Network stability: If every server runs at rate $R_{\max}$ constantly, then there exist well-known scheduling algorithms ensuring a time-independent upper bound on the size of all queues, in the temporary sessions model, for any given bounded adversary and any network topology. Such algorithms are called *universally stable*. More specifically, it was shown in [1] that several scheduling algorithms – including Farthest-to-Go (FTG) and Nearest-to-Source (NTS), whose definitions we provide later – are universally stable, and also guarantee bounded end-to-end packet delay. By contrast, some very natural algorithms such as First-in-First-out (FIFO) and Last-in-First-out (LIFO) are not universally stable.

In the permanent sessions model, the most widely studied scheduling algorithm is Generalized Processor Sharing (GPS), in particular its packetized form that is sometimes called Weighted Fair Queueing (WFQ) [8]. Under GPS and WFQ, each server operates by splitting service among all backlogged connections according to some predetermined weights. We shall focus on the version known as Rate Proportional Processor Sharing (RPPS), in which the weight for connection i is equal to ρ_i at each server. Parekh and Gallager [13], [14] proved that if each server always runs at rate $R_{\max}$ then the packetized version of RPPS is stable and they derived an end-to-end delay bound for each connection. (See (2) in Section I-C3.) We note that RPPS highlights the difference between the two traffic models in question, since it has been established that RPPS is *not* necessarily stable in the temporary sessions model [3].

2) Energy Efficiency: The study of energy minimization via rate adaptation was initiated by Yao et al. [16], where they considered energy functions of the form $f(x) = x^\alpha$. Subsequently, a large number of papers focused on the problem of conserving energy on a single server. Most of this prior work falls in two categories. In the first one (e.g. [5], [7]), every job has an associated deadline, and the goal is to minimize energy while meeting all the deadlines. In the second category (e.g. [4], [15]), jobs do not have individual deadlines and the goal is to minimize the sum of the energy used plus the aggregate response time of the jobs. Note that even in the single-server case neither of these objectives directly addresses our goal of minimizing energy consumption while maintaining stability.

Another body of work focuses on the *powerdown* model, in which the servers cannot alter their processing rates but can toggle between the on and off states at a switching cost. For example, [10] discusses the energy consequences of putting router and switch components to sleep. The survey [11] presents known results for both the powerdown and the rate adaptation models.

As already mentioned, much less attention was paid to scheduling for energy minimization in *networks* of servers. Nedevschi et al. [12] considered both rate adaptation and powerdown in the context of multiple servers and concluded that energy could be saved if we batch together packets with the same source and destination. Alternative techniques for batching packets in the powerdown model and thereby minimizing the number of transmissions between states were also explored in [2]. To the best of our knowledge, no prior work has attempted to tackle the specific problem that we deal with here, namely that of minimizing energy usage while maintaining network stability.

C. Results

Our paper is divided into three main sections.

1) Single-server results: In Section II we focus on scheduling a single server in isolation. This allows us to study basic issues such as the tradeoff between queue sizes and energy

in this simplest setting. For example, suppose $\mathsf{opt}_B(t)$ is the optimal energy necessary to keep the queue size bounded by B up to time t. We show that no online algorithm can simultaneously keep the energy consumption within a certain factor of opt_B *and* the queue size within a corresponding constant factor of B. This establishes a concrete limit on what one can hope for in terms of maintaining stability while minimizing energy.

We then propose three rate-adaptive policies to set server rates: Batch, SlowStart and queue-based. Roughly speaking, the Batch policy accumulates data of size $2B$ before serving it at a rate equal to the average arrival rate of this data. This policy ensures a maximum queue size of $O\big(B\log_2\frac{R_{\max}}{R_{\min}}\big)$ and energy usage of $O(6^\alpha\cdot\mathsf{opt}_B(t))$. The SlowStart policy starts with rate $R_{\min}$ at the beginning of each interval where the queue is non-empty, and after some time it ramps up the processing rate linearly, in a deterministic fashion. SlowStart ensures a maximum queue size of $O\big(\sigma+B\frac{R_{\max}^2}{R_{\min}^2}\big)$ and energy consumption of $O(2^\alpha\cdot\mathsf{opt}_B(t))$. The queue-based policy (or rather family of policies) sets the server rate solely as a function of the current queue size. Although we cannot bound the energy usage of this approach for all admissible traffic, when the arrival rate is constant it can keep the queue size bounded by $O(B)$ while consuming $O(\mathsf{opt}_B(t))$ energy.

2) Results for Temporary Sessions Model: In Section IV we examine the multiple-server scenario in the temporary sessions model. Our starting point is the universally stable [1] algorithm Farthest-to-Go (FTG), which gives priority to data farthest from its destination (in terms of server hops). We generalize all three rate-adaptive policies mentioned above to multiple servers. When combined with FTG, both Batch and SlowStart yield the desired properties of bounded queue size and bounded energy consumption. Additionally, we derive a bound on end-to-end delay in the case of SlowStart, although we are so far unable to establish a similar claim for Batch.

Note that for "traditional" work-conserving scheduling algorithms, bounded delay is equivalent to bounded queue (hence stability). However, this is no longer true when rate adaptation is an option, as demonstrated by the example in Section III.

In the interest of space, our presentation focuses on combining SlowStart with FTG. All results in this section carry over if we replace FTG with another universally stable algorithm Nearest-to-Source (NTS), which gives priority to data closest to its source.

3) Results for Permanent Sessions Model: For the Permanent Sessions Model, our starting point is the work of Parekh and Gallager [13], [14], who showed that if every server always runs at $R_{\max}$, Weighted Fair Queueing guarantees the following end-to-end delay bound for each connection i,

$$\frac{\sigma_i+(K_i-1)L_i}{\rho_i}+K_i\frac{\max_i L_i}{R_{\max}} \tag{2}$$

where σ_i, ρ_i, K_i and L_i are respectively the burst size, connection rate, hop count and maximum packet size for connection i. Our major result in this section is that for any rate-adaptive policy that uses rates between $R_{\min}$ and $R_{\max}$ and always uses rate $R_{\max}$ when the queue exceeds a threshold U, if we schedule according to Weighted Fair Queueing then the end-to-end delay is bounded by

$$\frac{\sigma_i+(K_i-1)L_i}{\rho_i}+K_i\left(\frac{\max_i L_i}{R_{\min}}+\frac{R_{\max}}{R_{\min}}\frac{U}{\rho_i}\right).$$

II. The Single-server Case

We begin our analysis by focusing on a single server in isolation. This will allow us to determine some of the basic tradeoffs between queue size and energy usage. First of all, we define a simple lower bound on energy consumption to keep the queue size bounded by B, which shall be used as a benchmark henceforth. We then present a bound on the optimal tradeoff between queue size and energy efficiency that can be achieved by any rate adaptation policy. Subsequently, in Sections II-C, II-D, and II-E, we propose three approaches to set the server rate adaptively and show upper bounds on their energy usage and queue size.

A. Lower bound on energy usage

Recall that $\mathsf{opt}_B(t)$ is the minimum amount of energy required by time t if we wish to keep the queue bounded by B at all times. Let $\mathsf{opt}_B(t,t')$ be defined similarly on the interval $[t,t')$. We derive a simple bound on $\mathsf{opt}_B(t,t')$.

Lemma 1. *For a single server e,*

$$\mathsf{opt}_B(t,t')\geq f\left(\max\left\{R_{\min},\frac{A_e(t,t')-B}{t'-t}\right\}\right)\cdot(t'-t).$$

Proof: Since data of size $A_e(t,t')$ arrives during the time interval $[t,t')$ and the queue size is to be less than B at time t', then the amount of data that must be processed during $[t,t')$ is at least $\max\{0,A_e(t,t')-B\}$. However, we are assuming that $f(\cdot)$ is a convex function and the minimum processing rate is $R_{\min}$, thus in order to serve that amount of data with minimal energy usage the server must operate at speed $\max\big\{R_{\min},(A_e(t,t')-B)/(t'-t)\big\}$ throughout the interval $[t,t')$. The bound follows. ∎

We shall assume that $B\geq\sigma$, since one cannot hope to maintain the queue size smaller than the burst size. Even then, the energy bound of Lemma 1 may not be achievable, for instance if most of the data injected during $[t,t')$ is injected towards the end of the interval.

B. Bound on tradeoff between queue size and energy

Intuitively, guaranteeing a better bound on queue size should incur higher energy usage. We establish that such a tradeoff is inherently unavoidable.

Lemma 2. *Let $x_1=\frac{B}{R_{\min}},x_2,x_3,\ldots$ be a sequence that satisfies*

$$x_j f\left(\frac{B}{2x_j}\right)\geq\nu\sum_{\ell<j}x_\ell f\left(\frac{B}{x_\ell}\right), \tag{3}$$

for some given ν. Further, suppose that a rate adaptation policy RA *uses energy at most $\nu\cdot\mathsf{opt}_B(t)$ by time t, for all times t. Then, the maximum queue size under* RA *is at least $(J(\nu)+1)B/2$, where $J(\nu)=\operatorname{argmax}\big\{j:x_j\geq\frac{B}{R_{\max}}\big\}$.*

Proof: We define a sequence $t_0 = 0, t_1, \ldots, t_{J(\nu)}$, with $t_j = t_{j-1} + x_j$ for $1 \le j \le J(\nu)$. Assume that data of size B arrives at each time $t_0, t_1, \ldots, t_{J(\nu)}$. An optimal rate adaptation policy would serve data of size B during each interval $[t_{j-1}, t_j)$ for $1 \le j \le J(\nu)$, so that the queue size remains at most B.

Nevertheless, the policy RA cannot follow this behavior, for the simple reason that it cannot predict the future. More precisely, when data of size B arrives at time t_j, $j \ge 1$, RA can deduce what the optimal schedule should have been for serving all data injected at $t_0, \ldots, t_{j-1}$. Then, by Lemma 1,

$$\mathsf{opt}_B(t_j) \ge \sum_{\ell \le j} (t_\ell - t_{\ell-1}) f\left(\frac{B}{t_\ell - t_{\ell-1}}\right) = \sum_{\ell \le j} x_\ell f\left(\frac{B}{x_\ell}\right) .$$

Since RA does not know whether (or when) additional data will be injected in the future, it can only afford to use at most $\nu \cdot \mathsf{opt}_B(t_j)$ energy in the interval $[0, t_{j+1})$, hence *a fortiori* in the interval $[t_j, t_{j+1})$. From (3),

$$\nu \cdot \mathsf{opt}_B(t_j) \le (t_{j+1} - t_j) f\left(\frac{B/2}{t_{j+1} - t_j}\right) ,$$

which implies that the amount of data RA can serve in $[t_j, t_{j+1})$ is no more than $B/2$. Finally, during the interval $[t_0, t_1)$, RA sets the speed to $R_{\min}$ by default and serves data of size B. Hence, at time $t_{J(\nu)}$ the queue size is at least $(J(\nu)+1)B - B - (J(\nu)-1)B/2 = (J(\nu)+1)B/2$. ∎

Corollary 3. *Suppose that the energy function has the form $f(s) = s^\alpha$ and that a rate adaptation policy* RA *uses energy at most $\nu \cdot \mathsf{opt}_B(t)$ by time t, for all times t. Then, the maximum queue size under* RA *is at least $\Omega\big(B \log_\nu(R_{\max}/R_{\min})\big)$.*

Sketch of proof: For $j = 1, 2, \ldots,$ let

$$x_j = \frac{B}{R_{\min}} \left(2^\alpha \nu + 1\right)^{\frac{j-1}{1-\alpha}} ,$$

$$\text{and} \quad J(\nu) = \left\lfloor (\alpha - 1) \log_{2^\alpha \nu + 1} \frac{R_{\max}}{R_{\min}} + 1 \right\rfloor .$$

Algebraic manipulation shows that the above definition of x_i satisfies (3). ∎

C. The Batch policy

We now show how to ensure both stability and near-optimal energy consumption using a rate adaptation policy that we call Batch. The basic idea is to wait until just enough data has arrived at the server so that our lower bound on $\mathsf{opt}_B(t)$ allows for a transmission rate that can serve all this data. Moreover, in the interest of simplifying the analysis, let us first assume that the server may operate even at rates higher than $R_{\max}$.

To begin with, define a *busy interval* as a time interval during which the queue is non-empty. Suppose that a busy interval I starts at time τ_0, and let $\tau_j = \min\{t \in I \mid A_e(\tau_0, t) \ge 2Bj\}$. Therefore, in each interval $[\tau_j, \tau_{j+1})$ data of size $2B$ arrives at the queue, and the policy makes sure that an equivalent amount of data is served by time $2\tau_{j+1} - \tau_j$. This is achieved by setting $r_e(t) = \max\left\{R_{\min}, \sum 2B/(\tau_{j+1} - \tau_j)\right\}$, where the summation is over all indices j such that $t \in [\tau_{j+1}, 2\tau_{j+1} - \tau_j)$. Denote these indices (if any such exist) by $j_1 < j_2 < \cdots < j_m$, and the interval $[\tau_{j_m}, \tau_{j_m+1})$ by $I(t)$. We have:

Lemma 4. *If $r_e(t) > R_{\min}$, then $r_e(t) \le 6B/|I(t)|$.*

Proof: For a given t, note that $r_e(t) > R_{\min}$ guarantees the existence of indices $j_1, j_2, \ldots, j_m$ as defined above, for some $m \ge 1$, and thus also the existence of $I(t)$. Since $j_\ell \ge j_{\ell-1} + 1$, observe that $\tau_{j_\ell} \ge \tau_{j_{\ell-1}+1}$. Furthermore, $t \le 2\tau_{j_\ell+1} - \tau_{j_\ell}$ implies $t - \tau_{j_\ell+1} \le \tau_{j_\ell+1} - \tau_{j_\ell}$. Combining the above yields $t - \tau_{j_\ell+1} \le \tau_{j_\ell+1} - \tau_{j_{\ell-1}+1}$, and hence $2(t - \tau_{j_\ell+1}) \le t - \tau_{j_{\ell-1}+1}$. Consequently, $r_e(t)$ equals

$$\begin{aligned}
\sum_{\ell=1}^{m} \frac{2B}{\tau_{j_\ell+1} - \tau_{j_\ell}} &= \sum_{\ell=1}^{m-1} \frac{2B}{\tau_{j_\ell+1} - \tau_{j_\ell}} + \frac{2B}{\tau_{j_m+1} - \tau_{j_m}} \\
&\le \sum_{\ell=1}^{m-1} \frac{2B}{t - \tau_{j_\ell+1}} + \frac{2B}{\tau_{j_m+1} - \tau_{j_m}} \\
&\le \sum_{\ell=1}^{m-1} \frac{2^{\ell-m+1} 2B}{t - \tau_{j_{m-1}+1}} + \frac{2B}{\tau_{j_m+1} - \tau_{j_m}} \\
&\le \frac{4B}{t - \tau_{j_{m-1}+1}} + \frac{2B}{\tau_{j_m+1} - \tau_{j_m}} \\
&\le \frac{4B}{\tau_{j_m+1} - \tau_{j_m}} + \frac{2B}{\tau_{j_m+1} - \tau_{j_m}} \\
&= \frac{6B}{\tau_{j_m+1} - \tau_{j_m}} = \frac{6B}{|I(t)|} ,
\end{aligned}$$

where the last inequality is due to the fact that $t - \tau_{j_{m-1}+1} \ge t - \tau_{j_m} \ge \tau_{j_m+1} - \tau_{j_m}$. ∎

Lemma 5. *The total energy used by* Batch *up until time t is at most $\sup_s \frac{f(6s)}{f(s)} \cdot \mathsf{opt}_B(t)$. For $f(s) = s^\alpha$, this is at most $6^\alpha \cdot \mathsf{opt}_B(t)$.*

Proof: Clearly, we need only consider the energy consumption of Batch during busy intervals – or, even more restrictively, during intervals where $r_e(t) > R_{\min}$. That holds because $R_{\min}$ is the most energy-efficient among the allowable rates, i.e. it minimizes the ratio $f(s)/s$, owing to our assumption about the energy function in Section I-A.

Within a busy period, consider some interval $[\tau_j, \tau_{j+1})$ and let $I'(\tau_j) = \{t \mid I(t) = [\tau_j, \tau_{j+1})\}$. Obviously $|I'(\tau_j)| \le \tau_{j+1} - \tau_j$, by the definition of $I(t)$. Due to Lemmas 4 and 1, the energy consumption during $I'(\tau_j)$ is at most

$$\begin{aligned}
f\left(\frac{6B}{\tau_{j+1} - \tau_j}\right) \cdot I'(\tau_j) &\le f\left(\frac{6B}{\tau_{j+1} - \tau_j}\right) \cdot (\tau_{j+1} - \tau_j) \\
&\le \sup_s \frac{f(6s)}{f(s)} \cdot \mathsf{opt}_B(\tau_j, \tau_{j+1}) .
\end{aligned}$$

Repeating this for every interval of the form $[\tau_j, \tau_{j+1})$, across all busy periods, accounts for the energy consumption of all intervals where $r_e(t) > R_{\min}$, without overlaps. The lemma is thus established. ∎

Subsequently, we derive a bound on the queue size.

Lemma 6. *The rate adaptation policy* Batch *guarantees that the queue size is never larger than $2B\big(\log_2(R_{\max}/R_{\min}) + 4\big)$.*

Sketch of proof: At time t, the amount of data still in the queue is given by

$$\sum_{\ell=1}^{m} 2B \cdot \frac{2\tau_{j_\ell+1} - \tau_{j_\ell} - t}{\tau_{j_\ell+1} - \tau_{j_\ell}} \le 2B \cdot m \ .$$

We need to bound m. On one hand, it is fairly straightforward to deduce that $t - \tau_{j_1+1} \le \tau_{j_1+1} - \tau_{j_1} \le 4B/R_{\min}$, otherwise the busy interval would end before τ_{j_1+1}. On the other hand, $t - \tau_{j_{m-1}+1} \ge \tau_{j_m+1} - \tau_{j_m} \ge B/R_{\max}$, because we assumed that $B \ge \sigma$. Together with $2(t-\tau_{j_\ell+1}) \le t-\tau_{j_{\ell-1}+1}$, the above yield $m \le \log_2(R_{\max}/R_{\min}) + 4$, as required. ∎

Finally, we explain how Batch works on a server whose maximum speed is strictly $R_{\max}$, which is the case of practical interest. More specifically, Batch tries to simulate its own behavior on an unrestricted server, which we described earlier. This involves two modes of operation, *normal* and *catch-up*. While in normal mode, the policy sets the same rate as it would set on the simulated unrestricted server. As soon as the latter rate exceeds $R_{\max}$, Batch enters catch-up mode, in which the rate is held at $R_{\max}$ constantly, and only reverts to normal mode when the queue sizes of the two servers, actual and simulated, coincide.

Naturally, in normal mode the bounds of Lemmas 5 and 6 remain valid. Suppose that just before switching to catch-up mode, the queue size is $Q \le 2B\big(\log_2(R_{\max}/R_{\min})+4\big)$. After remaining in that mode for a time interval of length λ, the queue size becomes $\le Q+\sigma+(1-\epsilon)R_{\max}\lambda - R_{\max}\lambda \le Q+\sigma$, which also implies that Batch will not stay in catch-up mode for ever. Furthermore, during the entire interval spent in catch-up mode, the actual server processed (at constant rate) exactly the same amount of data as the simulated one (at variable rate). Since $f(\cdot)$ is convex, the former server consumed no more energy than the latter.

Theorem 7. *The rate adaptation policy* Batch *ensures that maximum queue size is bounded by* $\Omega\big(B\log(R_{\max}/R_{\min})\big)$ *and consumes at most constant times the optimal energy.*

D. The SlowStart policy

In addition to Batch, let us introduce another policy named SlowStart. It stipulates that $r_e(t) = \min\{R_{\max}, g(t-\theta_e(t))\}$, where $g(\cdot)$ is a simple piecewise linear function

$$g(x) = \begin{cases} R_{\min} & \text{if } 0 \le x \le \frac{2B}{R_{\min}} \\ \frac{R_{\min}^2}{2B} \cdot x & \text{if } x > \frac{2B}{R_{\min}} \end{cases} ,$$

and $\theta_e(t)$ denotes the beginning of the busy interval of e that contains t, if one such exists, else is equal to t.

Within a busy interval, note that SlowStart initially sets the speed at $R_{\min}$ and later increases it linearly (hence the name), in a deterministic fashion. Interestingly, traffic arrivals have no direct effect on the speed, other than by determining when the busy interval ends. Below we summarize the properties of SlowStart in the single-server setting.

Theorem 8. *The rate adaptation policy* SlowStart *ensures that maximum queue size is bounded by* $\Omega\big(BR_{\max}^2/R_{\min}^2\big)$ *and consumes at most* $\sup_s \frac{f(2s)}{f(s)} \cdot \mathsf{opt}_B(t)$ *energy up until time* t. *For* $f(s) = s^\alpha$, *this is at most* $2^\alpha \cdot \mathsf{opt}_B(t)$.

Sketch of proof: Essentially a special case of Theorems 15 and 16 of Section IV, for $n = d = 1$. ∎

Remark. Although the above queue bound is worse than that of Batch, the SlowStart policy is nevertheless valuable, as we employ a variant of it in Section IV.

E. Queue-based policies

A queue-based policy sets the server rate according to $r_e(t) = r(q_e(t))$, where $q_e(t)$ is the queue size at time t, and $r(\cdot)$ is a non-negative and non-decreasing continuous function. In particular, if the queue size exceeds some threshold U then $r(q_e(t))$ is set to the maximum rate $R_{\max}$. Otherwise, $r(q_e(t))$ is at least the minimum rate $R_{\min}$. If traffic arrives at a constant rate, it is easy to provide good bounds for both queue and energy under such a policy.

Theorem 9. *Suppose that traffic arrives at the server at a constant rate* $\rho \le (1-\varepsilon)R_{\max}$. *Under a queue-based rate adaptation policy for which* $U = B$, *the queue size never exceeds* B *and the energy consumption up to time* t *is* $O(\mathsf{opt}_B(t))$, *for large enough* t.

Proof: Starting from an empty queue at time 0, both queue size and server speed begin to increase, until the speed reaches ρ. At that time, the queue size is at most U, because of the monotonicity of $r(\cdot)$ and the fact that $\rho < R_{\max}$. Both the server speed and queue size remain constant thereafter. Additionally, the energy usage up to time t is at most $t \cdot f(\rho)$, whereas $\mathsf{opt}_B(t) \ge t \cdot f\big((\rho t - B)/t\big)$. ∎

For arbitrary admissible traffic, it is easy to see that the above queue bound increases to $B + \sigma$, however we do not know whether any energy bound can be guaranteed.

III. Bounded Queue vs. Bounded Delay

We now turn to a network of multiple servers. Before diving into more complex results, let us state a simple but perhaps somewhat unexpected observation on the relationship between bounded queues and bounded delays. A folklore result states the equivalence between bounded queue and bounded delay when server rates are kept at $R_{\max}$.

Theorem 10 (Folklore)**.** *If a work-conserving algorithm always works at* $R_{\max}$, *then bounded delay is equivalent to bounded queue under any bounded adversary of rate* $(\sigma, 1-\varepsilon)$.

However, the above theorem no longer holds for some rate-adaptive algorithms. In particular, a stable rate-adaptive algorithm may not guarantee a bounded delay for every packet.

Lemma 11. *Neither* FTG *nor* NTS *guarantees a finite packet delay even in the permanent sessions model.*

Sketch of proof: Consider FTG on the following configuration. The network consists of three servers e_1, e_2, e_3 on a line and carries two connections: connection 1 goes to e_3 through e_1 and e_2, with rate $(1-2\varepsilon)R_{\max}$, and connection 2

goes to e_2 through e_1, with rate $\varepsilon R_{\max}$. Note that packets of connection 1 in the queue of e_1 have priority over those of connection 2, under FTG.

Suppose that at some point the queue of e_1 contains packets of both connections, and has size q° such that $r(q^\circ) \leq (1-2\varepsilon)R_{\max}$. Then, if connection-1 packets are henceforth injected at rate $r(q^\circ)$ and no more connection-2 packets are injected, existing connection-2 packets may be held *indefinitely* in e_1, even though its queue size remains bounded by q°. ■

IV. Temporary Sessions

As in Section II-E, consider a queue-based rate adaptation policy with threshold U. If $r(q_e(t)) < R_{\max}$ at time t, we say that all packets in the queue of server e are *withheld* in e at t. Naturally, a packet p may be withheld in e for two or more noncontiguous time intervals.

Lemma 12. *Denote by $w_e(t)$ the total size of packets in the queue of e at time t that are (or will be) withheld in e at* any *finite time $\geq t$. We have $w_e(t) \leq U$.*

Proof: To the contrary, suppose that $w_e(t) > U$ and let $t^* \geq t$ be the earliest time at which any of these packets is withheld in e. At that time, $q_e(t^*) > w_e(t) > U$, since none of said packets could have been processed until then. Consequently, $r(q_e(t^*)) = R_{\max}$ and hence no packet should be withheld at t^*, which is a contradiction. ■

Theorem 13. *Under any queue-based rate adaptation policy with finite threshold U,* FTG *and* NTS *are universally stable.*

Proof: Consider FTG; the proof for NTS is symmetric. We argue that having adversary $\mathcal{A}$ inject traffic in the rate-adaptive network G can be *quasi-simulated* by another adversary $\mathcal{A}'$ injecting traffic in a non-rate-adaptive system G', as described below. More precisely, we claim that it is within the capabilities of $\mathcal{A}'$ to ensure that at every time t the difference between the number of packets in each network is at most nU, where n is the number of servers of G.

The network G' is derived from G via the following construction: for each server e of G create d new servers $\pi_1(e), \pi_2(e), \ldots, \pi_d(e)$, and the resulting network is G'. All servers of G' follow the FTG algorithm as well. Furthermore, if $\mathcal{A}$ is a bounded adversary of rate $(\sigma, 1-\varepsilon)$, then $\mathcal{A}'$ is of rate $(\sigma + nU, 1-\varepsilon)$ and also knows the behavior of $\mathcal{A}$.

We now explain how the quasi-simulation takes place. Consider a packet p injected into G at time t, to be routed over consecutive servers $e_1, e_2, \ldots, e_k$, with $1 \leq k \leq d$. If p is not withheld anywhere until it is absorbed, then $\mathcal{A}'$ injects a copy of p into G' at t, with the same routing. Observe that $\mathcal{A}'$ can know what happens to p in the future, since the rate-adaptive network G is deterministic. On the other hand, suppose that p is withheld in $\ell \leq k$ distinct servers $e_{i_1}, e_{i_2}, \ldots, e_{i_\ell}$, such that $1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$. For $1 \leq j \leq \ell$, denote by t_{i_j} the first time that p is withheld in e_{i_j} and by $\bar{t}_{i_j}$ the time that it is processed. In that case, $\mathcal{A}'$ injects a total of $\ell + 1$ copies of p – not all at the same time – with mutually disjoint routing paths. The first copy is injected at t, with routing $e_1, \ldots, e_{i_1-1}, \pi_1(e_{i_1}), \ldots, \pi_{d-i_1+1}(e_{i_1})$. Subsequently, for $2 \leq j \leq \ell$, the j^{th} copy is injected at time $\bar{t}_{i_{j-1}}$, with routing $e_{i_{j-1}}, \ldots, e_{i_j-1}, \pi_1(e_{i_j}), \ldots, \pi_{d-i_j+1}(e_{i_j})$. Finally, the last copy is injected at time $\bar{t}_{i_\ell}$, with routing $e_{i_\ell}, \ldots, e_k$.

Recall that in the interval $[t_0, t)$, $\mathcal{A}$ is allowed to inject packets of total size up to $\sigma + (1-\varepsilon)R_{\max}(t-t_0)$ whose routing includes server e of G, whereas the analogous bound for $\mathcal{A}'$ is greater by nU. By Lemma 12, this suffices to allow $\mathcal{A}'$ to inject any subsequent copies of packets whose respective first copies were injected *before* time t_0. (Note that if only packets originally injected at or after t_0 had to be considered, a bound of $\sigma + (1-\varepsilon)R_{\max}(t-t_0)$ would be enough.) Additionally, it is straightforward to realize that at any time step t, the queue size of server e in G is exactly equal to $w_e(t)$ plus the size of the corresponding queue in G', which is bounded thanks to the universal stability of FTG [1]. ■

Remark. The quasi-simulation technique employed in the above proof cannot be used to obtain bounded delay guarantees for individual packets, thus it does not contradict Lemma 11.

A. *Combining* SlowStart *and* Batch *with* FTG

In order to combine the rate adaptation policy SlowStart with the scheduling algorithm FTG, we transform it so that it classifies packets in a similar way as FTG does, i.e. by the remaining distance to their destination. First, we provide a few definitions. Denote by $X_{e,i}(t)$ the total size of packets in the queue of server e at time t that are $\geq i$ hops away from their respective destinations. Moreover, define $k_i = 0$ for $i > d$ and $k_i = n\big(k_{i+1} + \sigma + 2R_{\max}^2(k_{i+1}+B)/R_{\min}^2\big)$ for $1 \leq i \leq d$.

For $t \geq 0$, the (revised) policy SlowStart sets the rate of e to $r_e(t) = \min\big\{R_{\max}, \max_{1\leq i\leq d} g_i\big(t - \theta_{e,i}(t)\big)\big\}$, where

$$g_i(x) = \begin{cases} R_{\min} & \text{if } 0 \leq x \leq \frac{2(k_{i+1}+B)}{R_{\min}} \\ \frac{R_{\min}^2}{2(k_{i+1}+B)} \cdot x & \text{if } x > \frac{2(k_{i+1}+B)}{R_{\min}} \end{cases}$$

and $\theta_{e,i}(t) = \sup\{t' \leq t \mid X_{e,i}(t') = 0\}$. Again, observe that each $g_i(\cdot)$ is a simple, piecewise linear function. Furthermore, the following property is readily verified:

Proposition 14. *For every $1 \leq i \leq d$ and every $x > 0$,*

$$g_i(x) = \max\left\{R_{\min}, 2\frac{\int_0^x g_i(u)\,\mathrm{d}u - k_{i+1} - B}{x}\right\}. \quad (4)$$

Theorem 15. *Combined with the rate adaptation policy* SlowStart*,* FTG *is universally stable and guarantees bounded packet delay.*

Proof: Via backwards induction on i, we argue that $\sum_e X_{e,i}(t) \leq k_i$ and

$$|X_{e,i}(t)| \leq k_{i+1} - \varepsilon R_{\max}\big(t - \theta_{e,i}(t)\big) + \sigma + \\ + 2R_{\max}^2(k_{i+1}+B)/R_{\min}^2 \quad (5)$$

for all $t \geq 0$, all $i \geq 1$, and every server e. These hold trivially for $i > d$, since the routing path of any packet contains $\leq d$

hops and thus $X_{e,i}(t) = 0$. Now, suppose that both inequalities are also valid for $i = \eta+1$, where $1 \le \eta \le d$. Then, for $i = \eta$, the packets in e at time t with $\ge \eta$ hops remaining may be partitioned in two categories: (i) packets that were injected at time $\theta_{e,\eta}(t)$ or later, whose total size is at most $\sigma + (1-\varepsilon)R_{\max}\big(t - \theta_{e,\eta}(t)\big)$, and (ii) packets that already existed in the system *before* $\theta_{e,\eta}(t)$, in queues *other than* that of e and still having $\ge \eta+1$ hops to go at that time, whose total size is at most $k_{\eta+1}$. Further, note that from $\theta_{e,\eta}(t)$ until t the queue processes at least $R_{\max}\big(t-\theta_{e,\eta}(t)-2R_{\max}(k_{\eta+1}+B)/R_{\min}^2\big)$ packets. Hence, $X_{e,\eta}(t) \le k_{\eta+1} - \varepsilon R_{\max}\big(t-\theta_{e,\eta}(t)\big) + \sigma + 2R_{\max}^2(k_{\eta+1}+B)/R_{\min}^2$ and $\sum_e X_{e,\eta}(t) \le n\big(k_{\eta+1} + \sigma + 2R_{\max}^2(k_{\eta+1}+B)/R_{\min}^2\big) = k_\eta$, completing the induction.

Moreover, (5) implies that $t - \theta_{e,i}(t) \le k_i/(n\varepsilon R_{\max})$. Therefore, a packet that arrives at the queue of e and still has $\ge i$ hops to go will be processed within $k_i/(n\varepsilon R_{\max})$ time. This means that every packet reaches its destination at most $\sum_{i=1}^d k_i/(n\varepsilon R_{\max})$ time after its injection. Finally, we deduce that the total number of packets in the system at any time is $\le k_1$. ∎

Theorem 16. *Combined with the rate adaptation policy* SlowStart*,* FTG *consumes at most* $d \cdot \sup_s \frac{f(2s)}{f(s)} \cdot \mathsf{opt}_B(t)$ *energy up until time* t*, where* d *is the maximum hop count of routing paths. For* $f(s) = s^\alpha$*, this is at most* $d \cdot 2^\alpha \cdot \mathsf{opt}_B(t)$*.*

Remark. In a multiple-server context, the meaning of $\mathsf{opt}_B(t)$ is slightly different than the one in Section II-A. Specifically, the desired bound B pertains to the sum of (i) the queue size of a server e at any given moment, and (ii) the total size of all packets that are elsewhere in the network at that time and must go through e in the future to arrive at their respective destinations. In all other respects, the statement and validity of Lemma 1 remain unaffected.

Proof of Theorem 16: Our analysis is done on a per-server basis. As usual, we need only account for energy usage in time intervals where $r_e(t) > R_{\min}$. First, define $\zeta_{e,i}(t) = \inf\{t' \ge t \mid X_{e,i}(t') = 0\}$ and $I_i(t) = \big[\theta_{e,i}(t), \zeta_{e,i}(t)\big)$. Obviously $t \in I_i(t)$, and if $t_1 \ne t_2$ then $I_i(t_1)$ and $I_i(t_2)$ either coincide or are mutually disjoint.

Furthermore, for $1 \le i \le d$ let $T_i = \{t \in [0,t) \mid r_e(t) = g_i(t - \theta_{e,i}(t)) > R_{\min}\}$ and $\mathcal{I}_i = \{I_i(t) \cap [0,t) \mid t \in T_i\}$. Now, fix i and consider each interval $I \in \mathcal{I}_i$. If $t_a = \inf I$ and $t_b = \sup I$, then we have

$$\begin{aligned}
\int_{I\cap T_i} f(r_e(u))\,\mathrm{d}u &= \int_{I\cap T_i} f(g_i(u-t_a))\,\mathrm{d}u \\
&\le f(g_i(t_b - t_a)) \cdot |I \cap T_i| \\
&\le f\left(2\frac{\int_{t_a}^{t_b} g_i(u)\,\mathrm{d}u - k_{i+1} - B}{t_b - t_a}\right) \cdot |I| \\
&\le f\left(2\frac{A_e(t_a,t_b) - B}{t_b - t_a}\right) \cdot (t_b - t_a) \\
&\le \sup_s \frac{f(2s)}{f(s)} \cdot \mathsf{opt}_B(t_a,t_b)\ ,
\end{aligned}$$

where the first inequality above holds because $g_i(t-t_a)$ is non-decreasing in I, the second because of (4) plus the fact that $|I \cap T_i| \le |I|$, while the third is due to arguments similar to those that established (5). Since the intervals contained in each $\mathcal{I}_i$ are mutually disjoint, we straightforwardly deduce that the total energy consumed by e up until time t is at most

$$\begin{aligned}
\sum_{i=1}^d \left(\int_{T_i} f(r_e(u))\,\mathrm{d}u\right) &= \sum_{i=1}^d \left(\sum_{I\in\mathcal{I}_i}\left(\int_{I\cap T_i} f(r_e(u))\,\mathrm{d}u\right)\right) \\
&\le \sum_{i=1}^d \left(\sup_s \frac{f(2s)}{f(s)} \cdot \mathsf{opt}_B(t)\right)\ ,
\end{aligned}$$

which directly implies the theorem. ∎

Last but not least, it is rather straightforward to modify Batch along the same lines as we did with SlowStart, for the purpose of combining it with FTG and obtaining a stable rate-adaptive algorithm with near-optimal energy consumption. In fact, the queue bound thus derived is better than that of the SlowStart-FTG combination, consistently with the advantage of Batch over SlowStart in the single-server case. On the other hand, we could not derive a guarantee on end-to-end packet delay for the Batch-FTG algorithm, for reasons similar to those discussed in Section III.

Due to space constraints, we omit a detailed description and analysis of the aforementioned algorithm. Nevertheless, its salient properties are summed up in the following theorem.

Theorem 17. *Combined with the rate adaptation policy* Batch*,* FTG *is universally stable and consumes at most* $d \cdot \sup_s \frac{f(6s)}{f(s)} \cdot \mathsf{opt}_B(t)$ *energy up until time* t*. For* $f(s) = s^\alpha$*, this is at most* $d \cdot 6^\alpha \cdot \mathsf{opt}_B(t)$*.*

V. Permanent Sessions

At this point, we turn our attention to the permanent sessions model, in which traffic is injected into fixed-path connections. Recall that σ_i is the burst size for connection i and ρ_i is the injection rate. Hence, if $H_i(t,t')$ is the amount of traffic injected into connection i during the time interval $[t,t')$, then $H_i(t,t') \le \sigma_i + \rho_i(t'-t)$. Let S_e be the set of connections passing through server e, let $q_{i,e}(t)$ be the amount of connection i data queued at server e at time t, and let $q_e(t) = \sum_{i\in S_e} q_{i,e}(t)$.

Consider a traditional setting in which each server e always runs at the maximum speed $R_{\max}$. The most commonly studied algorithm is Generalized Processor Sharing and its packetized version Weighted Fair Queueing (WFQ) [8], [14]. At all times GPS[2] splits its service equally among its backlogged sessions in proportion to the connection rates ρ_i. That is, at time t connection i is served at rate $R_{\max}\phi_i(t)$ where $\phi_i(t) := \rho_i \big/ \big(\sum_{j\in S_e, q_{j,e}(t)>0} \rho_j\big)$. GPS is an idealized algorithm that assumes that service can be divided evenly among multiple sessions. For this reason Weighted Fair Queueing was introduced, which serves data on a packet-by-packet basis.

[2]Strictly speaking, the version of GPS called Rate Proportional Processor Sharing (RPPS).

WFQ runs a virtual GPS scheduler in the background and whenever it has to serve a packet it chooses the packet that would be scheduled next according to the GPS scheduler, assuming that no more packets arrive.

We now define rate-adaptive (RA) versions of GPS and WFQ that we call RA-GPS and RA-WFQ respectively. At all times t, server e operates at a speed $r(q_e(t))$. We focus on rate adaptation in the following generic form. If $q_e(t)$ exceeds a certain threshold U, then e operates at the full rate $R_{\max}$; otherwise, as long as the queue is non-empty, e operates at a rate no lower than the minimum rate $R_{\min}$. At time t any backlogged connection i is served at rate $r(q_e(t))\phi_i(t)$. RA-WFQ operates an RA-GPS scheduler in the background and always runs the server at the same speed as RA-GPS. Whenever it has to serve a packet it chooses the packet that would be scheduled next according to the RA-GPS scheduler, assuming that no more packets arrive.

Note that even though $q_{i,e}(t)$ may have different values depending on whether we use RA-GPS or RA-WFQ, both the queue size $q_e(t)$ and the server processing rate $r(q_e(t))$ are independent of whichever of the two algorithms is used.

The classic analysis of Parekh and Gallager [14] showed that if $r(q) = R_{\max}$ for all q then the end-to-end delay bound for connection i under WFQ is given by,

$$\frac{\sigma_i + (K_i - 1)L_i}{\rho_i} + K_i \frac{L_{\max}}{R_{\max}} , \tag{6}$$

where K_i is the number of hops on the path of connection i, L_i is the size of the maximum packet injected into connection i, and $L_{\max} = \max_i L_i$. The aim of this section is derive a similar bound for RA-WFQ. Our analysis in motivated by a derivation of the bound (6) presented in the book [6]. In particular, we show:

Theorem 18. *The end-to-end delay for connection-i packets under* RA-WFQ *is bounded by,*

$$\frac{\sigma_i + (K_i - 1)L_i}{\rho_i} + K_i \left(\frac{L_{\max}}{R_{\min}} + \frac{R_{\max}}{R_{\min}} \frac{U}{\rho_i} \right) .$$

Proof: Let us first concentrate on connection i at one particular server e. For the nth packet arrival from connection i at link e, let a_n be its arrival time at e, and let ℓ_n be its packet size. By the definition of GPS, ρ_i is a lower bound on the service rate to connection i. (Note that $\sum_{j \in S_e} \rho_i \le R_{\max}$ by assumption.) Moreover, define a sequence

$$h_n = \max\{a_n, h_{n-1}\} + \frac{\ell_n}{\rho_i} .$$

Following [6], we say that a server is a (ρ_i, y) Guaranteed Rate ((ρ_i, y)-GR) server if the finishing time of the nth packet is at most $h_n + y$ for all n. It is shown in [6] that under GPS server e is a $(\rho_i, 0)$-GR server for connection i.

Let g_n^{RA} be the finishing time for the nth packet under RA-GPS. In the following, we bound the difference between h_n and g_n^{RA}.

Lemma 19. $g_n^{\mathsf{RA}} - h_n \le \frac{R_{\max}}{R_{\min}} \frac{U}{\rho_i}$. *Hence, under* RA-GPS *server e is a $\left(\rho_i, \frac{R_{\max}}{R_{\min}} \frac{U}{\rho_i}\right)$-GR server for connection i.*

Proof: It is not hard to see that if connection i is always served at rate exactly ρ_i then the finishing time of the nth packet will be exactly h_n. We examine $s_i(t)$, the cumulative amount of service connection i receives by time t if it is always offered server ρ_i. The slope of $s_i(t)$ is either ρ_i or 0. (The latter happens when the connection-i queue is empty.) We define $s_i^{\mathsf{RA}}(t)$ to be the cumulative service connection i receives under RA-GPS. We now claim that $s_i(t) - s_i^{\mathsf{RA}}(t) \le U$ for any t. This is because whenever $s_i(t) - s_i^{\mathsf{RA}}(t)$ reaches U it must be the case the value of $q_{i,e}(t)$ under RA-GPS is at least U. This in turn implies that $q_e(t) \ge U$ and so $r(q_e(t)) = R_{\max}$. Hence RA-GPS serves connection i at rate at least ρ_i and so $s_i(t) - s_i^{\mathsf{RA}}(t)$ cannot increase.

The above argument implies that $s_i(h_n) - s_i^{\mathsf{RA}}(h_n) \le U$. Since connection-$i$ is always served at rate at least $R_{\min}\rho_i / R_{\max}$, even under RA-GPS, this in turn implies that $g_n^{\mathsf{RA}} \le h_n + \frac{R_{\max} U}{R_{\min}\rho_i}$, as required. ∎

We now bound the difference in finishing time for a packet at server e under RA-GPS and RA-WFQ. The intuition here is that processing a connection-i packet under RA-WFQ can be delayed if a packet from a different connection just started being processed. The worst-case delay happens when the other packet is large in size and is processed at the lowest possible rate.

Lemma 20. *The finish time for a packet under* RA-WFQ *is at most its finish time under* RA-GPS *plus* $\frac{L_{\max}}{R_{\min}}$.

Sketch of proof: Entirely analogous to the corresponding result for WFQ in [6]. Omitted here for reasons of space. ∎

Corollary 21. *Under* RA-WFQ*, server e is a $\left(\rho_i, \frac{R_{\max} U}{R_{\min}\rho_i} + \frac{L_{\max}}{R_{\min}}\right)$-GR server for connection i.*

Subsequently, we focus on concatenating a sequence of GR links, using a concatenation theorem from [6].

Theorem 22 (Concatenation Theorem [6])**.** *The concatenation of M (x_j, y_j)-GR servers, where $1 \le j \le M$, results in an (x, y)-GR server where $x = \min_{1 \le j \le M} x_j$ and $y = \sum_{1 \le j \le M-1} (y_j + L_i / x_j)$.*

Suppose connection i goes through K_i links. We know that each link e along the path is (x_e, y_e)-GR, where x_e and y_e are defined by Corollary 21. Applying Theorem 22 to Corollary 21 yields:

Lemma 23. *For the K_i links along the path for connection i, concatenating these K_i links results in an (x_i, y_i)-GR server, where $x_i = \rho_i$ and $y_i = K_i\left(\frac{R_{\max} U}{R_{\min}\rho_i} + \frac{L_{\max}}{R_{\min}}\right) + (K_i - 1)\frac{L_i}{\rho_i}$.*

Next, we appeal to the following result from [6] to obtain an end-to-end delay bound for RA-WFQ.

Lemma 24 (Delay Bound [6])**.** *For a (σ, ρ) traffic source, a (ρ, y)-GR server guarantees a delay of $\frac{\sigma}{\rho} + y$.*

Combining Lemmas 23 and 24, we derive that the end-to-end delay for connection i under RA-WFQ is upper bounded

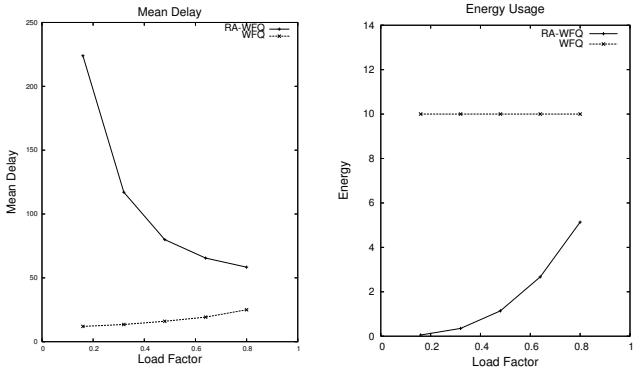

Fig. 1. The delay and energy performance of WFQ and RA-WFQ.

by

$$\frac{\sigma_i + (K_i - 1)L_i}{\rho_i} + K_i \left(\frac{L_{\max}}{R_{\min}} + \frac{R_{\max}}{R_{\min}} \frac{U}{\rho_i} \right),$$

as required. ∎

A. Simulation

We now briefly describe simulation results that illustrate the tradeoff between delay and energy usage. For our experiments, we use a home-grown simulator written in C. The network topology is a linear array of 10 identical servers. Moreover, the network supports one long connection passing through all 10 servers, and 10 short connections each passing through one of the servers. We normalize the maximum processing rate $R_{\max}$ of each server to 1, and consider the energy function $f(s) = s^3$. The injection rate of the long connection is $\ell/4$, for some load factor ℓ that we vary between 0.16 and 0.8. The injection rate of each short connection is $3\ell/4$. Hence, the load on each server is ℓ. Observe that our study is explicitly scale-invariant: time is specified in time slots, injection and processing rates are specified as a fraction of $R_{\max}$, and power consumption is specified as a function of the processing rate.

The left-hand plot of Figure 1 shows the average delay of packets under WFQ and RA-WFQ, where the latter uses the queue-based rate adaptation policy given by $r(q) = \min\{q/10, 1\}$. Similarly, the right-hand plot shows energy usage. As expected, RA-WFQ exhibits significantly lower energy consumption for light loads, at the expense of higher packet delays.

VI. Conclusion

In this paper we studied energy-aware scheduling algorithms with the objective of achieving bounded queue sizes (which implies network stability) and bounded delays, while simultaneously minimizing energy usage. We proposed several policies, with varying degrees of complexity, to adjust the server rates in response to incoming traffic. By combining these policies with existing stable algorithms, such as Farthest-to-Go and Weighted Fair Queueing, we were able to achieve the aforementioned objective and also derive different tradeoffs between queue sizes/delays and energy consumption.

Several interesting open questions remain. For example, consider Longest-in-System (LIS), which gives priority to packets that have been in the network for the longest time; thus, it may be regarded as a "global" FIFO. Assuming that servers run at $R_{\max}$ all the time, LIS is known to be universally stable [1] and has many desirable properties. Consequently, inventing a rate-adaptive version of LIS with near-optimal energy usage, while also preserving its salient characteristics, would be a substantial achievement.

Finally, on a more general level, it would be very interesting to know whether there is a single rate-adaptive approach that can be applied to *any* stable scheduling algorithm to guarantee bounded queue size and near-optimal energy consumption.

Acknowledgement: This R&D work was completed with the support of the U.S. Department of Energy (DOE), award no. DE-EE0002887. However, any opinions, findings, conclusions and recommendations expressed herein are those of the authors and do not necessarily reflect the views of the DOE.

References

[1] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results and performance bounds for greedy contention-resolution protocols. *Journal of the ACM*, 48(1):39–69, January 2001.

[2] M. Andrews, A. Fernandez Anta, L. Zhang, and W. Zhao. Routing and scheduling for energy and delay minimization in the powerdown model. In *IEEE INFOCOM '10*, March 2010.

[3] M. Andrews and L. Zhang. The effects of temporary sessions on network performance. In *SODA '00: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457, San Francisco, CA, USA, January 2000.

[4] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *SODA '09: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 693–701, Philadelphia, PA, USA, 2009.

[5] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM*, 54(1), 2007.

[6] J.-Y. Le Boudec and P. Thiran. *Network Calculus*. Springer Verlag, http://ica1www.epfl.ch/PS_files/NetCal.htm, 2004.

[7] H.-L. Chan, W.-T. Chan, T. W. Lam, L.-K. Lee, K.-S. Mak, and P. W. H. Wong. Energy efficient online deadline scheduling. In N. Bansal, K. Pruhs, and C. Stein, editors, *SODA '07*, pages 795–804, 2007.

[8] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking: Research and Experience*, 1:3–26, 1990.

[9] Enhanced Intel SpeedStep technology for the Intel Pentium M processor. Intel White Paper 301170-001, 2004.

[10] M. Gupta and S. Singh. Greening of the Internet. In *SIGCOMM '03*, pages 19–26, New York, NY, USA, 2003.

[11] S. Irani and K. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.

[12] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In J. Crowcroft and M. Dahlin, editors, *NSDI '08*, pages 323–336, 2008.

[13] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, 1993.

[14] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, 1994.

[15] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. In *IEEE INFOCOM '09*, pages 2007–2015, 2009.

[16] F. F. Yao, A. J. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *FOCS '95*, pages 374–382, 1995.