

# $E^2$ WFQ: An Energy Efficient Fair Scheduling Policy For Wireless Systems

Vijay Raghunathan, Saurabh Ganeriwal, Curt Schurgers, and Mani Srivastava

Networked and Embedded Systems Lab (NESL)  
Department of Electrical Engineering  
University of California, Los Angeles

{vijay, saurabh, curts, mbs}@ee.ucla.edu

## ABSTRACT

As embedded systems are being networked, often wirelessly, an increasingly larger share of their total energy budget is due to the communication. This necessitates the development of power management techniques that address communication subsystems, such as radios, as opposed to computation subsystems, such as embedded processors, to which most of the research effort thus far has been devoted. In this paper, we present  $E^2$ WFQ, an energy efficient version of the Weighted Fair Queuing (WFQ) algorithm for packet scheduling in communication systems. We employ a recently proposed radio power management technique, Dynamic Modulation Scaling (DMS), as a control knob to enable energy-latency tradeoffs during wireless packet scheduling. The use of  $E^2$ WFQ results in an energy aware packet scheduler, which exploits the statistics of the input arrival pattern as well as the variability in packet lengths. Simulation results show that large savings in energy consumption can be obtained through the use of our scheduling scheme, compared to conventional WFQ, with only a small, bounded increase in worst case packet latency.

## 1. INTRODUCTION

Conventional low power design techniques [1, 2] and hardware architectures [3] only target digital computation systems, and relatively little work has been done for power optimization of the wireless communication subsystem. In many wireless embedded systems, communication energy dominates the energy consumed for computation [4], accentuating the need for radio power management methodologies. For example, in the wireless sensor nodes from Rockwell Inc. [5], transmitting one bit consumes 1500 to 2700 times [4] (depending on the transmission range) as much energy as executing one instruction. Therefore, in wireless devices, power management cannot just be limited to computation subsystems, such as processors, but has to be extended to communication subsystems, such as radios, as well.

Radio power management is, however, complex, since the dependence of radio energy consumption on supply voltage and other

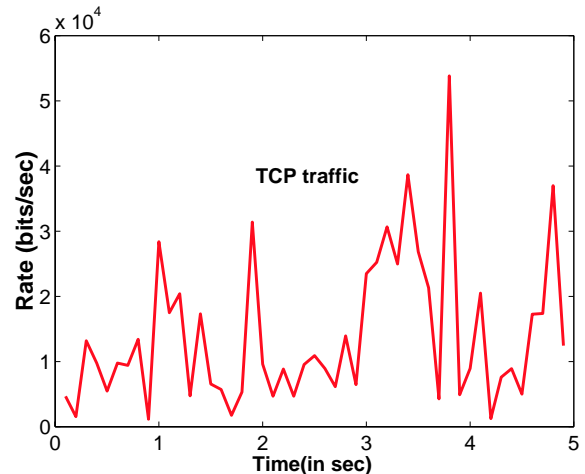


Figure 1: A 5 second workload trace at a network router

circuit parameters is weak. Existing power management techniques that are effective for computation subsystems, such as Dynamic Voltage Scaling (DVS) [6], therefore do not result in the required energy savings. However, there exist similar control knobs on the radio that can be exploited for power management. The recently proposed technique of Dynamic Modulation Scaling (DMS) [7] uses the modulation level as an energy-speed control knob that can be fine-tuned to enable dynamic power-performance tradeoffs in the communication system (similar to what DVS provides for digital circuits [6]).

As is the case with variable voltage computation systems, significant energy benefits can be achieved in wireless communication systems by recognizing that peak performance (*i.e.*, service rate) is not always required. Since network traffic is characterized by a time varying workload requirement, energy can be saved by dynamically adapting the output transmission rate accordingly, through the use of DMS. Figure 1 shows a 5 second snapshot of a real workload trace for a TCP traffic stream at a network router [8], exemplifying the inherent workload variability. For effective system-level power management, we need to develop algorithms that can exploit this variability by using control knobs such as DMS.

### 1.1 Paper contributions

In this paper, we present an energy aware version of the Weighted Fair Queuing (WFQ) [9] scheduling policy for communication systems, which we call  $E^2$ WFQ. Our algorithm results in an energy

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISLPED'02, August 12-14, 2002, Monterey, California, USA.  
Copyright 2002 ACM 1-58113-475-4/02/0008 ...\$5.00.

aware packet scheduler, which is capable of operating at hitherto unreachable points on the energy-latency tradeoff curve. Our previous work [10] showed the energy benefits of using DMS in the context of a deadline based real-time scheduling scheme. However, WFQ based packet schedulers are far more widely used than deadline based packet schedulers (both in wired and wireless environments) due to their desirable properties, which are elaborated further in Section 2.1. Therefore, the techniques presented in this paper find applicability in enhancing the energy awareness of a much larger class of communication systems. To the best of our knowledge, this is the first work that attempts to incorporate energy awareness into rate based fair scheduling.

## 1.2 Related work

Several power management techniques have been proposed for the energy efficient design and operation of computation subsystems. One of the most effective techniques is DVS [6], where workload variability is exploited for energy savings by dynamically adjusting the processor's supply voltage and clock frequency to match the instantaneous performance requirement. Numerous energy aware task scheduling schemes have also been proposed, which utilize DVS to yield significant energy savings. Variable voltage task scheduling for base station like environments has been explored in [11, 12], while the work described in [13, 14, 15, 16, 17, 18] deals with energy aware real-time task scheduling.

Unlike the large body of work that exists on variable voltage task scheduling, relatively little work has been done for energy efficient wireless packet scheduling. Fair packet scheduling has been an active research topic in the networking community for a long time. Several fair scheduling schemes have been proposed for both wired (e.g., Weighted Round Robin [19], Start-Time Fair Queuing [20], Worst-Case Fair Weighted Fair Queuing [21]) and wireless (e.g., Channel-State Independent Wireless Fair Queuing [22], Wireless Packet Service [23], Server Based Fairness Approach [24]) environments. However, all of them are based on the concept of Generalized Processor Sharing [25], and its packetized version, Packet-by-Packet Generalized Processor Sharing [26] (PGPS, also known as WFQ [9]). Since network traffic usually displays a high temporal variation, the leaky bucket mechanism [25] is used as a standard way to regulate input streams, and control their extent of variability. Finally, low power medium access protocols based on packet scheduling for wireless ATM networks were proposed in [27].

## 2. BACKGROUND

### 2.1 Generalized Processor Sharing (GPS)

Since GPS is different from conventional CPU task scheduling disciplines, we first review some of its basic concepts. A GPS scheduler divides the total link capacity  $C$ , among  $N$  input streams according to their service requirements. Each stream  $i$  is characterized by a weight  $\phi_i$ , such that its service rate,  $g_i$ , is guaranteed to be [25]<sup>1</sup>:

$$g_i = \frac{\phi_i}{\sum_{j=1}^N \phi_j} \times C \quad (1)$$

GPS has the following attractive features: (i) Different input streams are well-isolated from each other (since each of them is

<sup>1</sup>Equation (1) assumes that all streams are backlogged (i.e., have packets waiting to be sent). If a stream is not backlogged, it is not given any share of the link. Further, it does not contribute to the summation in Equation (1). Thus, GPS divides the output link capacity among only those streams that are backlogged.

allocated a guaranteed rate). Therefore, a malicious input stream cannot starve other streams of link bandwidth by generating large amounts of traffic. (ii) By varying the  $\phi_i$ 's, we have the flexibility of changing the fraction of the output link bandwidth that is allocated to each stream. As long as the combined average input rate of all the streams is less than  $C$ , any positive assignment of  $\phi_i$ 's yields a stable system.

### 2.2 Realizable implementations of GPS: WFQ

The GPS service model is an ideal one in which the traffic is considered to be infinitely divisible, and all streams are served simultaneously. Although GPS cannot be realized in practice, several real implementations of packet schedulers (for wired, as well as wireless environments) are based on it, and try to emulate the service provided by GPS as closely as possible.

Let  $F_p$  be the time at which a packet would complete service under GPS. Then, a good approximation of GPS is a scheme that serves packets with earliest  $F_p$  first. This scheme was proposed independently by two groups of researchers as Weighted Fair Queuing [9], and Packetized GPS [26], respectively. The difference in packet delays between GPS and WFQ was shown to be bounded by  $\frac{L_{max}}{C}$ , where  $L_{max}$  is the maximum packet size, and  $C$  is the output rate of the system [26].

### 2.3 Leaky bucket mechanism

The leaky bucket mechanism is often used to regulate and police the traffic in a network. This model is attractive since it restricts the incoming traffic in terms of average rate, as well as burstiness. For each stream  $i$ , the amount of traffic,  $A_i(\tau, t)$  that enters the network in time interval  $(\tau, t]$  conforms to [25]:

$$A_i(\tau, t) \leq \sigma_i + \lambda_i \times (t - \tau), \forall t \geq \tau \geq 0 \quad (2)$$

where  $\lambda_i$  is a parameter characterizing the average rate of stream  $i$ , and  $\sigma_i$  is a parameter characterizing its burstiness. When the input sources are constrained by leaky buckets, it is possible to give a worst case queuing delay guarantee using GPS (and therefore, WFQ).

**Theorem 1:** *If the input traffic of stream  $i$  is constrained using a leaky bucket with parameters  $(\sigma_i, \lambda_i)$ , and  $g_i$  is its guaranteed rate, then the maximum delay for a packet of stream  $i$ , under GPS scheduling is given by [26]:*

$$D_i \leq \frac{\sigma_i}{g_i} \quad (3)$$

### 2.4 Dynamic Modulation Scaling

We next review the basics of DMS, which was introduced in [7]. To transmit information, bits are coded into channel symbols. The number of bits per symbol is given by the modulation level  $b$ . This  $b$  is the radio control knob that allows DMS to trade off energy versus delay. The average time to transmit one bit is given by Equation (4), where  $R_S$  is the symbol rate in number of symbols sent over the channel per second.

$$T_{bit} = \frac{1}{b \times R_S} \quad (4)$$

Although DMS is applicable to other scalable modulation schemes as well, we focus on Quadrature Amplitude Modulation (QAM) as it is both efficient and easy to implement [28]. The energy consumed for transmitting one bit is given by [7]:

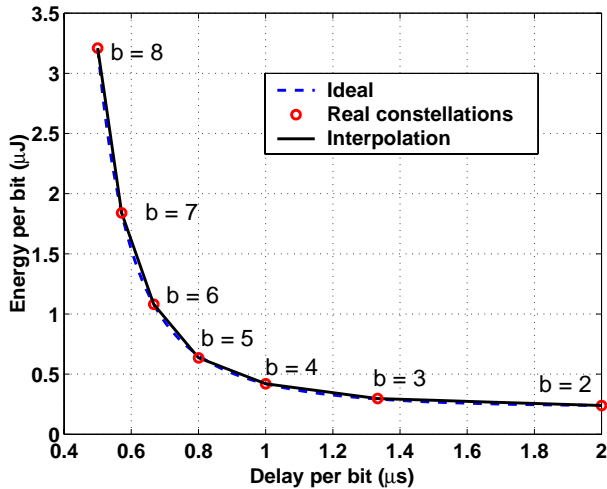


Figure 2: Energy-Delay tradeoff obtained through DMS

$$E_{bit} = C_S \times \frac{2^b - 1}{b} + C_E \times \frac{1}{b} \quad (5)$$

The first term gives the energy consumed in the radio front-end for generating the electro-magnetic waves that carry the information. Parameter  $C_S$  depends on the radio implementation, the wireless channel, the transmit distance, and the required error performance. Strictly speaking, it is also a function of  $b$ , but a very weak one [7], and therefore, we assume it to be a constant. The rest of the radio energy consumption is lumped into the second term of Equation (5), where  $C_E$  depends on the radio implementation. Although Equation (5) is only exact for *even* integer values of  $b$ , it is also a reasonable approximation when  $b$  is *odd*. In addition, a packet can be split into two parts, each with a different modulation. In this case, the average energy and delay per bit for the packet as a whole are a linear interpolation between the corresponding values of the two modulation levels.

Figure 2 plots the energy versus delay for the following parameter values:  $R_S = 250$  KHz,  $C_S = 100$  nJ, and  $C_E = 180$  nJ. The values of  $C_S$  and  $C_E$  are extracted from [29], which describes the implementation of an adaptive QAM system<sup>2</sup>. Figure 2 shows the operating points that correspond to real constellations and those that are obtained through interpolation. The curve labeled *Ideal* is calculated from the above equations. Each modulation scheme has a  $b_{min}$ , which is equal to 2 for QAM. The maximum modulation  $b_{max}$  is only bounded by implementation constraints. In our work, we choose  $b_{max}$  equal to 8, and scale the modulation level with a granularity of 0.5 bits/symbol. When the sender changes the modulation, the receiver needs to be told. To avoid a complicated sender-receiver protocol for modulation changes, and to limit the associated overhead, we restrict these changes to be done only at the start of packet transmissions. This finite time-granularity is a crucial difference between DMS and dynamic voltage scaling [6].

### 3. THE $E^2$ WFQ ALGORITHM

Having explained the basics of rate based fair scheduling and DMS, we next present our algorithm,  $E^2$ WFQ, which uses DMS to incorporate energy awareness into rate based scheduling. Figure 3

<sup>2</sup>Since the system in [29] was designed for high speed rather than low power applications, it is likely that these numbers can be reduced further through the use of dedicated circuit design techniques.

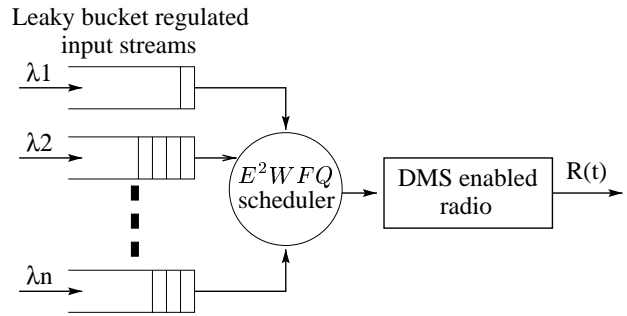


Figure 3:  $E^2$ WFQ scheduler serving leaky bucket regulated streams

shows a generic block diagram of our system, which consists of an  $E^2$ WFQ scheduler followed by a radio that can perform DMS. The scheduling technique can be used either for multiple streams sharing a point-to-point link, or for multiple streams destined to different one-hop neighbors of a wireless node (*e.g.*, a base station sending data to multiple clients). Each stream  $i$  is regulated by a leaky bucket with parameters  $(\sigma_i, \lambda_i)$ . Therefore, each of the  $N$  input streams produces packets at an average rate of  $\lambda_i$ . The average rate at which packets arrive at the scheduler is  $\lambda = \sum_{i=1}^N \lambda_i$ . The adaptive modulation radio ensures that the scheduler operates at a time varying output rate of  $R(t)$ . Stream  $i$  is allocated a weight  $\phi_i$ , which leads to a corresponding guaranteed rate,  $g_i$ .

#### 3.1 Energy saving opportunities

In many practical scenarios, the average input rate of a stream,  $\lambda_i$ , is lower than its guaranteed rate,  $g_i$ , resulting in a low link utilization. In addition, packet lengths may often be smaller than the maximum value, thereby reducing the utilization even further. Therefore, it is possible to operate at an instantaneous output rate,  $R(t)$  that is lower than the maximum rate,  $C$ , for most of the time. So, instead of operating at  $C$ , and shutting down the radio when idle, we slow transmissions down to a rate  $R(t)$ . As can be seen in Figure 2, the energy-delay curve is convex, which implies that slowing down the radio is more energy efficient than shutdown (similar to what is observed in DVS [6]).

#### 3.2 Overview of the problem

The goal of  $E^2$ WFQ is to adapt the instantaneous output rate  $R(t)$  to match the instantaneous workload. At the same time, we would like to bound the performance impact that may result. Due to the convexity of the energy-speed curve and Jensen's inequality ( $\overline{E(\tau)} \leq E(\overline{\tau})$ ), workload averaging results in higher energy savings. In fact, maximum energy savings would be obtained if we operated at the long term average input rate, thereby smoothing out all the input workload variations. However, buffering the input variations leads to a performance penalty due to an increase in packet delays. Therefore, the crux of the problem reduces to determining the degree of buffering (*i.e.*, how much workload averaging to perform) while still bounding the increase in packet delays.

#### 3.3 Monitoring the input rate

We observe that the instantaneous queue size (*i.e.*, number of packets in the queue) is a good indicator of the instantaneous arrival rate. If the input rate suddenly becomes greater than the output rate, the queue size increases. On the other hand, if the input rate suddenly drops below the output rate, the queue gets drained. We introduce a new parameter  $\Delta$ , which determines the system's

sponse to workload variations, by deciding the degree of buffering. As we will see shortly,  $\Delta$  also determines the maximum impact that our scheme can have on packet delay.

**Definition 1:** We define  $\Delta$  to be the desired time from a packet's arrival at the end of the queue to its departure from the head of the queue.

We can see that if the input rate remains constant,  $\Delta$  is the delay experienced by every packet in the queue.

### 3.4 Computing the required output rate

Whenever a packet arrives, the required rate of the particular stream is recomputed. Assume that the newly arrived packet is the  $m$ 'th packet in the queue, and the current time is  $\tau$ . Let the arrival times of the  $m$  packets in the queue be given by  $A_1, A_2, \dots, A_m$  (where  $A_m = \tau$ ). Then, the required rate ( $r_{i,k}$ ) for the  $k$ 'th packet to have a total delay of  $\Delta$  is given by:

$$r_{i,k} = \frac{k \times L_i}{(A_k + \Delta - \tau)} \quad \forall k \in \{1, \dots, m\} \quad (6)$$

where  $L_i$  is the length of a packet belonging to stream  $i$ . Two observations are apparent from Equation (6), (i) The required rate for the newly arrived packet (i.e., last packet in the queue) is  $\frac{m \times L_i}{\Delta}$ , and (ii) If the output rate is equal to the required rate, then the required rate of this packet remains constant as it progresses through the queue.

The required rate for a stream  $R_{out,i}$  is the minimum rate at which Equation (6) is satisfied for all the  $m$  packets in the queue. However, to ensure isolation of input streams, this required rate cannot be greater than  $g_i$ . Therefore, we have:

$$R_{out,i} = \text{Min} \{ \text{Max} (r_{i,1}, r_{i,2}, \dots, r_{i,m}), g_i \} \quad (7)$$

Summing the required rate over all the streams, we get the instantaneous required rate of the system to be  $R = \sum_{i=1}^N R_{out,i}$ .

### 3.5 Setting the output link speed

Although the maximum output link rate is  $C$ , the instantaneous required rate by all the streams together is only  $R = \sum_{i=1}^N R_{out,i}$ . This means that the output link can be slowed down to just meet the instantaneous requirement, thus saving energy. The new modulation level for the outgoing packets is given by:

$$b_{instantaneous} = \frac{R}{C} \times b_{max} \quad (8)$$

### 3.6 Accounting for variable packet sizes

Quite often, packet lengths in network traffic can be variable. This is especially true for packets generated by multimedia streams such as variable bit rate audio and video codecs. To exploit the variation in packet lengths to further scale down the modulation level and obtain more energy savings, we generalize the calculation of a packet's required rate. If the length of the  $k$ 'th packet in the queue is  $L_{i,k}$ , and there are a total of  $m$  packets in the queue, then the following equation is used to compute the required rate of a packet, instead of Equation (6):

$$r_{i,k} = \frac{\sum_{j=1}^k L_{i,j}}{(A_k + \Delta - \tau)} \quad \forall k \in \{1, \dots, m\} \quad (9)$$

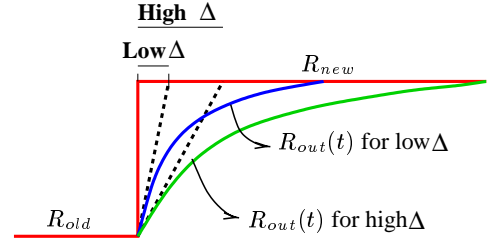


Figure 4: The effect of  $\Delta$  on the output rate

### 3.7 Delay guarantee provided by $E^2$ WFQ

Through the choice of the parameter  $\Delta$ , our scheduling scheme provides the following delay bound for packets.

**Theorem 2:** The maximum delay of a packet of stream  $i$ , under the  $E^2$ WFQ scheduling scheme is given by:

$$D_i^* \leq \left( D_i + \Delta + \frac{L_{max}}{C_{min}} \right) \leq \left( \frac{\sigma_i}{g_i} + \Delta + \frac{L_{max}}{C_{min}} \right) \quad (10)$$

where  $C_{min}$  is the output link capacity at a modulation level  $b_{min}$ . **Proof:** The proof follows directly from Theorem 1, the bounded difference in packet delays between GPS and WFQ, and the definition of  $\Delta$ .

### 3.8 Analyzing the system's response

We next analyze the system's response to a change in workload to highlight the effect of  $\Delta$ . Let the input rate for stream  $i$  be equal to  $R_{old}$ . If we set the output rate to be equal to  $R_{old}$ , then the queue soon reaches a steady state, and the number of packets in the queue remains constant. Let the system be at such a steady state at time  $t$ , and the number of packets in the queue be  $y(t)$ . In this state, any packet that arrives, sees exactly  $y(t) - 1$  packets ahead of it in the queue, each of length, say  $L_i$ . All these packets (including the one that just arrived) have to complete transmission within a time  $\Delta$ . Therefore, the output rate,  $R_{out}(t)$  is given by (see first paragraph after Equation (6)):

$$R_{out}(t) = \frac{y(t) \times L_i}{\Delta} = R_{old} \quad (11)$$

Now, let the input rate increase abruptly to  $R_{new}$ . We want to analyze how fast the system stabilizes to its new steady state. After a time  $\delta t$ , the number of packets in the queue becomes:

$$y(t + \delta t) = y(t) + \frac{R_{new} - R_{out}(t)}{L_i} \times \delta t \quad (12)$$

The new output rate will be given by  $R_{out}(t + \delta t)$ , which after some simple substitutions, can be written as:

$$R_{out}(t + \delta t) = R_{out}(t) + \frac{R_{new} - R_{out}(t)}{\Delta} \times \delta t \quad (13)$$

Therefore, the rate at which the output rate changes is given by:

$$\frac{\delta R_{out}}{\delta t} = \frac{1}{\Delta} \times (R_{new} - R_{out}(t)) \quad (14)$$

Solving this differential equation, and applying the initial condition  $R_{out}(t) = R_{old}$ , we get:

$$R_{out}(t) = R_{new} + (R_{old} - R_{new}) \times e^{-\frac{t}{\Delta}} \quad (15)$$

The evolution of  $R_{out}(t)$  is shown in Figure 4. Therefore, it can be concluded that  $\Delta$  is the time constant of the first-order input

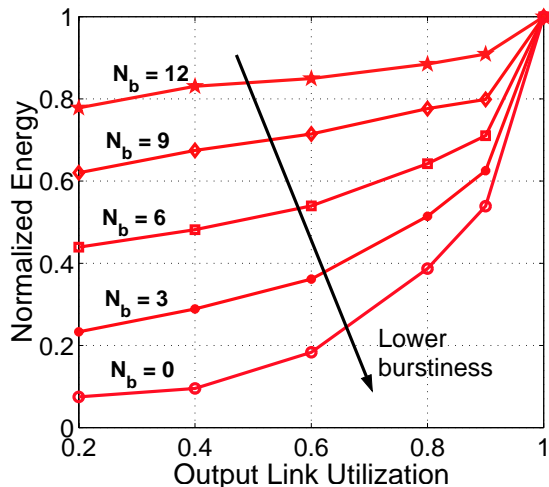


Figure 5: Normalized energy consumption of  $E^2WFQ$ , as a function of link utilization, for varying degrees of burstiness

response of the system. A high value of  $\Delta$  means that the system takes longer to switch to the new steady state, which implies that steep workload transients are filtered out. While this increases the energy savings, the maximum impact on packet delay also increases (see Equation (10)). On the other hand, a low value of  $\Delta$  means that the system is sensitive to changes in the input rate, and performs lesser workload averaging. This provides lower energy savings, but also results in a smaller impact on packet delay. The best choice of  $\Delta$  thus depends on the allowable delay, and the input rate variability. A detailed investigation of the optimal choice of  $\Delta$  is beyond the scope of this paper.

## 4. SIMULATION RESULTS

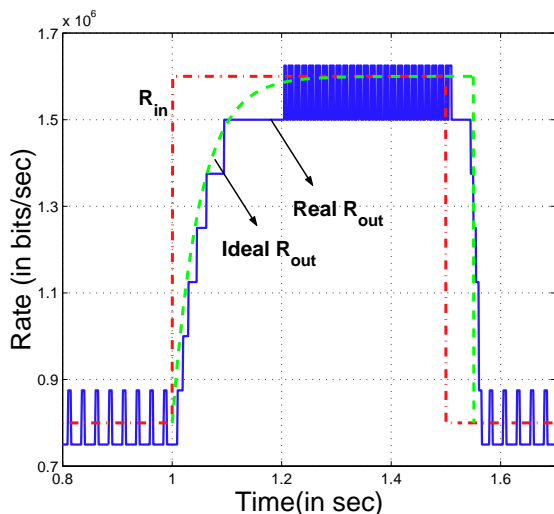


Figure 6: One second snapshot showing how the output rate follows the input rate

To evaluate the impact of our techniques, we carried out a number of simulations. In the following subsections, we describe our simulation framework, and present our results.

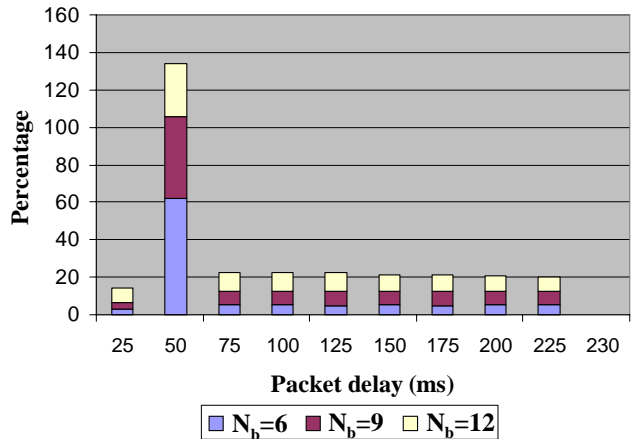


Figure 7: Packet delay histogram for three different levels of burstiness

### 4.1 Simulation framework

We used a C based discrete event simulator, PARSEC [30], and considered a network scenario where 4 streams share an output wireless link. The values of  $R_S$  and  $b_{max}$  are the same as in Section 2.4, resulting in a total link capacity  $C = 2$  Mbit/s. The streams are allocated equal weights such that each stream is guaranteed a rate of  $\frac{C}{4} = 500$  Kbit/sec. The maximum packet size is set to be  $L_{max} = 1000$  bits, and each stream is leaky bucket constrained with the same maximum burstiness parameter  $\sigma_i = 100$  packets. For our values,  $D_i$  is equal to 0.2 seconds. The average rate of the input is varied to change the link utilization. As explained in the previous section, the choice of  $\Delta$  offers a tradeoff between energy savings and queuing delay. In our work, we have set  $\Delta = 0.05$  seconds.

### 4.2 Discussion of the results

Figure 5 shows the energy consumed by  $E^2WFQ$ , normalized against the energy consumed by conventional WFQ, for varying values of output link utilization. As expected, the energy consumption of  $E^2WFQ$  decreases as the link utilization drops. Our algorithm reduces to conventional WFQ at a utilization of one, since there are no opportunities for slowing down. The different curves in Figure 5 show the energy consumption for varying degrees of burstiness (variability) in the input workload.  $N_b$  gives the number of equally spaced bursts appearing at a stream's input over the observation window of 1000 packets of the stream. A high value of  $N_b$  indicates a high degree of burstiness. For a given link utilization, increasing the level of input burstiness increases the energy consumption, which is intuitive, since bursts at the input cause the output rate to increase, thus decreasing the energy savings. Figure 6 shows how the output rate follows the input rate, for our choice of  $\Delta = 0.05$  seconds. The curve marked *Ideal R<sub>out</sub>* shows the curve predicted by Equation (15). The actual output rate can only increase in increments of  $\frac{L_i}{\Delta}$ , which is clearly seen in the curve marked *Real R<sub>out</sub>* in the figure.

Figure 7 shows the distribution of packet delays for varying levels of burstiness for a fixed link utilization. The bar at  $x = x_i$  indicates the percentage of packets whose delays lie in the interval  $(x_{i-1}, x_i]$ . There are three important observations that can be made from this figure: (i) Since the input rate fluctuates around its average rate, most of the packets experience a delay equal to  $\Delta$ . (ii) As the level of burstiness increases, the percentage of packets that experience a delay larger than  $\Delta$  also increases, and (iii) None of the

packets violate the delay bound given in Theorem 2. Our results (Figures 5 - 7) show that  $E^2W FQ$  achieves a significant reduction in energy consumption, at the cost of a relatively small increase in packet latencies.

## 5. CONCLUSIONS

In wireless embedded systems, communication energy is increasingly becoming the dominant component of total energy consumption. Power management techniques therefore, should also address communication subsystems such as radios, as opposed to only computation subsystems such as embedded processors. DMS is a technique that offers a power-speed control knob for the radio, and can thus be used for communication power management. Higher level techniques are needed that can exploit this control knob to enable system level energy-latency tradeoffs. In this paper, we have targeted the packet scheduling process and investigated avenues for making it energy aware. We have presented  $E^2W FQ$ , an energy efficient version of the Weighted Fair Queuing (WFQ) algorithm for packet scheduling in communication systems. Our algorithm results in an energy aware packet scheduler that provides significant energy savings (up to a factor of 10) with only a small increase in worst case packet delay.

## ACKNOWLEDGEMENTS

This paper is based in part on research funded through the DARPA PAC/C Program under AFRL Contract #F30602-00- C-0154, and through Semiconductor Research Corporation System Task Thrust ID 899.001.

## 6. REFERENCES

- [1] A. Raghunathan, N. K. Jha, and S. Dey, *High-level Power Analysis and Optimization*. Kluwer Academic Publishers, Norwell, MA, 1998.
- [2] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer Academic Publishers, Norwell, MA, 1997.
- [3] A. P. Chandrakasan and R. W. Brodersen, *Low Power CMOS Digital Design*. Kluwer Academic Publishers, Norwell, MA, 1996.
- [4] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy aware wireless microsensor networks", *IEEE Signal Processing Magazine*, vol. 19, iss. 2, pp. 40–50, March 2002.
- [5] Rockwell Inc. (<http://wins.rsc.rockwell.com>)
- [6] T. A. Pering, T. D. Burd, and R. W. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms", *Proc. ACM ISLPED*, pp. 76–81, 1998.
- [7] C. Schurgers, O. Aberthorne, and M. B. Srivastava, "Modulation scaling for energy aware communication systems", *Proc. ACM ISLPED*, pp. 96–99, 2001.
- [8] The Internet Traffic Archive (<http://ita.ee.lbl.gov>).
- [9] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm", *Internet Res. and Exper.*, vol. 1, 1990.
- [10] C. Schurgers, V. Raghunathan, and M. B. Srivastava, "Modulation scaling for real-time energy-aware packet scheduling", *Proc. IEEE GLOBECOM*, pp. 3653–3657, 2001.
- [11] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy", *Proc. First Symp. on OSDI*, pp. 13–23, 1994.
- [12] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU", *Proc. ACM MOBICOM*, pp. 13–25, 1995.
- [13] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors", *Proc. ACM ISLPED*, pp. 197–202, 1998.
- [14] I. Hong, M. Potkonjak, and M. B. Srivastava, "On-line scheduling of hard real-time tasks on variable voltage processors", *Proc. IEEE ICCAD*, pp. 653–656, 1998.
- [15] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems", *Proc. ACM DAC*, pp. 134–139, 1999.
- [16] V. Raghunathan, P. Spanos, and M. B. Srivastava, "Adaptive power-fidelity in energy-aware wireless systems", *Proc. IEEE RTSS*, pp. 106–115, 2001.
- [17] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low power embedded operating systems", *Proc. SOSP*, pp. 89–102, 2001.
- [18] F. Gruian, "Hard real-time scheduling for low energy using stochastic data and DVS processors", *Proc. ACM ISLPED*, pp. 46–51, 2001.
- [19] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin", *Proc. IEEE SIGCOMM*, pp. 231–242, 1995.
- [20] P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks", *Proc. IEEE SIGCOMM*, pp. 157–168, 1996.
- [21] J. C. R. Bennett and H. Zhang, "WF<sup>2</sup>Q: Worst-case fair weighted fair queueing", *Proc. IEEE INFOCOM*, pp. 120–128, 1996.
- [22] P. Lin, B. Bensaou, Q. L. Ding, and K. C. Chua, "A wireless fair scheduling algorithm for error-prone wireless channels", *Proc. ACM WoWMOM*, pp. 11–20, 2000.
- [23] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks", *Proc. ACM SIGCOMM*, pp. 63–74, 1997.
- [24] P. Ramanathan and P. Agrawal, "Adapting packet fair queueing algorithms to wireless networks", *Proc. ACM MOBICOM*, pp. 1–9, 1998.
- [25] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*. Addison-Wesley, 1997.
- [26] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case", *IEEE Trans. on Networking*, vol. 1, no. 3, pp. 344–357, June 1993.
- [27] K. M. Sivalingam, M. Srivastava, P. Agrawal, and J. C. Chen, "Low power access protocols based on scheduling for wireless and mobile ATM networks", *Proc. IEEE Universal Personal Communications Conference*, pp. 420–433, 1997.
- [28] J. G. Proakis, *Digital Communications*. McGraw Hill, 1989.
- [29] K. Cho and H. Samueli, "A 8.75-MBaud Single-Chip Digital QAM Modulator with Frequency-Agility and Beamforming Diversity", *Proc. IEEE Custom Integrated Circuits Conference*, pp. 27–30, 2000.
- [30] PARSEC parallel simulation language (<http://pcl.cs.ucla.edu/projects/parsec>)