A Power-Aware and QoS-Aware Service Model on Wireless Networks

Hao Zhu and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
E-mail: {hazhu, gcao}@cse.psu.edu

Abstract— **Many studies show that the wireless network interface (WNI) accounts for a significant part of the power consumed by mobile terminals. Thus, putting the WNI into sleep when it is idle is an effective technique to save power. To support streaming applications, existing techniques cannot put the WNI into sleep due to strict delay requirements. In this paper, we present a novel power-aware and QoS-aware service model over wireless networks. In the proposed model, mobile terminals use proxies to buffer data so that the WNIs can sleep for a long time period. To achieve power-aware communication while satisfying the delay requirement of each flow, a scheduling scheme, called *priority-based bulk scheduling* (PBS), is designed to decide which flow should be served at which time. Through analysis, we prove that the PBS service model can provide delay assurance and achieve power efficiency. We use Audio-on-Demand and Web access as case studies to evaluate the performance of the PBS service model. Experimental results show that PBS achieves excellent QoS provision for each flow and significantly reduces the power consumption.**

Index Terms: Power-aware, simulations, QoS, scheduling, buffer management, wireless networks.

I. Introduction

With the advent of the third generation wireless infrastructure, and the rapid growth of wireless communication technology, pervasive computing becomes possible: people with battery powered mobile terminals (cellular phones, PDAs, handheld computers, etc.) can access various kinds of services at any time any place. However, the goal of achieving ubiquitous connectivity with small-size and low-cost mobile terminals (MTs) is challenged by the power constraints. Most MTs are powered by battery, but the rate at which battery performance improves is fairly slow [18]. Aside from major breakthroughs, it is doubtful that significant improvement can be expected in the foreseeable future. Instead of trying to improve the amount of energy that can be packed into a power source, we can carefully design communication protocols so that the MTs can perform the same functions and provide the same services while minimizing their overall power consumption.

Understanding the power characteristics of the wireless network interface (WNI) used in MTs is important for designing power efficient communication protocols. A typical WNI may exist in *active* or *sleep* state. In the active state, the WNI may be in the transmit, receive and idle modes. Many studies [3], [12], [22], [24] show that the power consumed in the active state is similar, which is significantly higher than the power consumed in the sleep state. As a result, most of the work on power management concentrates on putting the WNI into sleep when it is idle. Stemm and Katz [22] studied transport layer approaches and application-driven approaches to help power down the WNI. Protocols [9], [10], [19], [21], [26] are proposed to put the WNI into sleep under the following conditions: the WNI is idle, the use of the WNI may collide with other MTs, or the use of the WNI suffers from interference. Most of these works focus on reducing the power consumption. This can be applied to most data applications which do not have QoS requirements, but may not be enough for streaming applications (e.g. audio/video-on-demand) due to lack of QoS provisions. Steaming applications have been very popular over the Internet, and it is becoming possible for mobile environments. For example, Nokia models from 5510 [15] start to support MP3. IDC [5] predicts that by 2006, in Europe alone, there will be 37 million users listen to music delivered through wireless channel. Compared to the power consumed on music playing, the WNI consumes much more energy (as much as 70% of the total power in Nokia 5510). Thus, to achieve Audio-on-Demand (AoD) in wireless networks, reducing the power consumption of the WNI is one of the major issue. Since streaming applications have QoS requirement, *how to achieve power saving without violating QoS* is a big challenge for supporting streaming applications on wireless networks.

It takes some time for a WNI to wake up from sleep. As reported in [20], the transition time from active to sleep and back to active is on the order of tens of milliseconds. Due to this state transition delay, existing power management schemes may not be directly applied to streaming applications since the inter-packet arrival time maybe too small and the MT cannot power off the WNI without violating the delay requirement. To deal with the transition delay, we propose to support power save mode using buffers. In the proposed solution, a proxy is added at the MT side. This proxy buffers enough data from the server, and lets the WNI enter sleep until the buffered data run out. Then, it wakes up the WNI and downloads more

This work was supported in part by the National Science Foundation (CAREER CCR-0092770 and ITR-0219711).

data to fill the buffer. Since a base station (BS) may serve a large number of MTs, simple solutions may not be able to manage the proxy without violating the QoS. For example, as a simple solution, the MT sends a request to the BS when its buffer is near empty, and the BS serves the MT's request until the MT's buffer is full and then serving another one. Although this approach can minimize the power consumption, it may violate the QoS when multiple MTs send requests at almost the same time. Under this condition, since the BS can only serve one MT at a time, some MTs may have to wait for a long time and the delay requirement may be violated.

In this paper, we propose a new scheduling scheme, called *priority-based bulk scheduling* (PBS) to utilize the client buffer to save power and provide QoS. Under PBS, the scheduler keeps track of the amount of prefetched (buffered) data for each flow. The flow with sufficient buffered data will be suspended until the buffered data runs out, whereas the flow with insufficient buffered data will be served based on its priority, which is determined by its delay requirement. With buffered data, the WNI can sleep long enough to offset the impact of the state transition delay. By suspending some flows, other active flows sharing the channel can obtain more bandwidth and take less time to fill the buffer. We also extend the service model to consider channel errors and applications that do not have strict delay requirements. Through analysis, we prove that PBS can provide delay assurance for real-time applications and is more power efficient than conventional rate-based fair queuing models. We use Audio-on-Demand and Web access as case studies to evaluate the performance of the PBS service model. Experimental results show that PBS achieves excellent QoS provision for each flow and significantly reduces the power consumption.

The rest of the paper is organized as follows. In the next section, we describe the system model. In Section III, we describe the details of the PBS service model. In Section IV, we evaluate its performance. Section V presents related work, and Section VI concludes the paper.

II. System Model

The geographical area is divided into cells in a wireless network. Inside each cell, the base station (BS) communicates with the mobile terminals (MTs) through uplink and downlink channels. Both uplink and downlink channels are divided into time slots. For uplink, we assume a channel can be either randomly accessed by MTs or granted through downlink (e.g., by reservation). Location-dependent errors may happen due to channel fading, interference, etc [13]. For simplicity, we assume the modulation techniques and coding techniques are fixed. Certainly, link adaptation schemes which can select the most suitable modulation and coding technique based on the current channel quality, can be complement to our solution. In order to achieve reliable transmission, each downlink packet must be acknowledged. The channel condition can be measured by the number of packet retransmissions.

Since a WNI spends significantly higher amount of power (from 10 to 100 times [3], [12], [22], [24]) during active compared to sleep, we use the accumulated WNI sleep time to measure the power efficiency of different schemes. In order to accurately measure the power consumption, we use the following notations.

- $T_{off \to on}$: time spent to transit from sleep to active.
- $T_{on \to off}$: time spent to transit from active to sleep.

We assume that the power consumed during the state transition from active to sleep, or from sleep to active is similar to that in the active mode, which is much larger than the power consumed in the sleep mode.

III. The PBS Service Model

In this section, we present the PBS service model. Before looking into details, let's first look at the drawbacks of existing scheduling algorithms in terms of power efficiency and QoS provision.

A. Background and Motivation

In order to provide guaranteed service over a shared link, several rate-based service disciplines have been proposed [25]. The principle of these service models is to provide each flow with a guaranteed data rate without being affected by other mis-behaving flows sharing the link. A scheduling algorithm can be classified as *work-conserving* or *non-work-conserving*. In the work-conserving scheduling, a server is never idle when there is a packet to send. In the non-work-conserving scheduling, a packet is not served until it is eligible [25], even though the server is idle at that time.

When applying the existing scheduling algorithms to wireless networks, power issues should be considered. As explained in Section I, putting the WNI into sleep is the most widely used method to save power. When work-conserving service discipline is used, it is difficult to put the WNI into sleep. The reason is as follows. When an MT shares the link with other MTs, if a work-conserving guaranteed service model is applied, the service sequence of the link depends on the scheduling pattern of all flows. The scheduling pattern is related to the number of backlogged flows and the deadlines of the head-of-line packets of these flows. Unfortunately, the MT does not know the following service sequence due to lack of global information regarding the scheduling pattern of all flows in the system. Thus, the WNI has to stay in active since it does not know when the next packet will arrive. This may cause the WNI to waste a lot of power. For example, as shown in Figure 1, suppose three flows

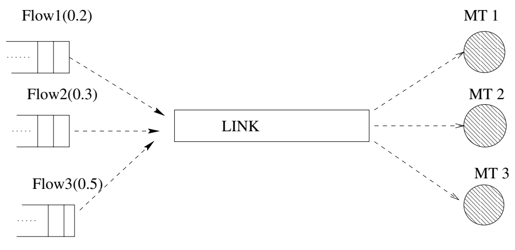


Fig. 1. An example of work-conserving link sharing

share a link of capacity C, and each flow has an MT as its receiver. Suppose all flows want to send B bits and their data rates are: $0.2C, 0.3C$ and $0.5C$ respectively. Under the work-conserving service model, if B is quite large, MT_1's active time is approximately $5B/C$, but MT_1's effective receive time is B/C. It is easy to see that almost 80% of the power has been wasted.

Non-work-conserving Virtual Clock (NVC): Non-work-conserving scheduling can be used to save power. The basic idea is to let the WNI enter sleep when it is not used. One simple approach is to let the BS and the MT mutually agree on a scheduling pattern. When the BS sends a packet to the MT, the scheduler piggybacks the information about the eligible time for the next packet to be transmitted. Note that the eligible time can be calculated based on the flow's data rate. The scheduler works in a *non-work-conserving* [25] manner since it will not serve the flow before the eligible time even though the channel is idle. Thus, the MT can enter sleep for a while until the eligible time of the next packet. This simple approach is referred to as the *Non-work-conserving Virtual Clock* (NVC) scheduling. Although NVC can save a large amount of power in theory, it may not be possible in practice. This is due to the reason that it takes some time and power for the WNI to transit between sleep and active. Suppose $T_{off\to on} = 10ms$, if the packet interval time, denoted by T_{intvl}, of the flow is less than $10ms$, we have $T_{on\to off} + T_{off\to on} > T_{intvl}$. Thus, the NVC scheme cannot put the WNI into sleep without violating the QoS requirement of the flow[1]. Even when $T_{on\to off} + T_{off\to on} < T_{intvl}$, not too much power can be saved unless $T_{on\to off} + T_{off\to on} \ll T_{intvl}$. From Figure

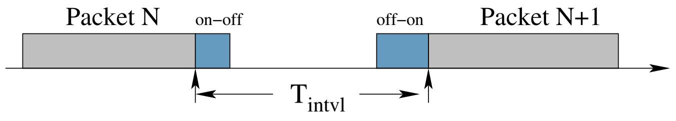


Fig. 2. The relationship between state transition and the packet interval time

2, we can see that $\frac{T_{on\to off}+T_{off\to on}}{T_{intvl}}$ of the total packet interval time will be used for state transition. In addition, since this transition also costs a lot of power, the NVC scheduling algorithm cannot save too much power unless $T_{on\to off} + T_{off\to on} \ll T_{intvl}$.

Bulk Scheduling (BKS): Based on the above reasoning, another approach, called *Bulk Scheduling (BKS)*, can be used to further reduce power. With this service policy, the channel is divided into bulk slots. A flow which needs to be served wakes up at the beginning of a bulk slot. The scheduler randomly selects a flow to serve at that time, and the selected flow will be served until the end of the bulk slot. Meanwhile, other losing flows enter sleep and wake up again to wait for the service at the beginning of the next bulk slot. Figure 3

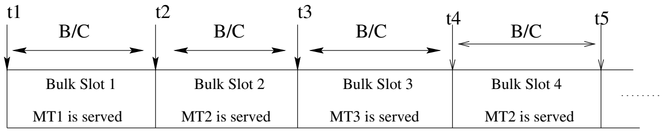


Fig. 3. An example of bulk scheduling

shows one example of bulk scheduling. In this example, three flows share the link and the receivers are MT_1, MT_2, and MT_3 respectively. Each bulk slot is equal to B/C, where B is the number of bits and C is the capacity of the link. For MT_3, it wakes up at t_1 and enters sleep since it found that the scheduler has selected MT_1 to serve. The same procedure happens at t_2. At t_3, it wakes up again and starts to receive data. Suppose MT_3 wants to transmit $k * B$ bits. If B is large enough so that the power used for state transitions between idle and active is negligible, the active time for MT_3 is only $k * B/C$, which allows MT_3 to stay in sleep for the maximum amount of time. Thus, if the bulk slot is significantly larger than $T_{on\to off} + T_{off\to on}$, the power consumption of state transition can be neglected, and bulk scheduling is an optimal service model in terms of power efficiency. However, bulk scheduling cannot provide QoS when multiple flows request data at the same time. For example, at t_1, suppose MT_3 and MT_1 will miss their deadline if waiting for another B/C time slot, scheduling MT_1 to serve will force MT_3 to miss its deadline.

In order to address the drawbacks of the NVC approach and the BKS approach, we propose a priority-based bulk scheduling (PBS) service model considering QoS provision and power efficiency. The basic idea of PBS is to let the MT buffer as much data as possible without affecting the QoS requirement of other flows. Relying on the buffered data, the MT can put its WNI into sleep and wake up only when the prefetched data is not enough to satisfy its QoS requirement. In this way, the WNI can power off for many time slots. Furthermore, as the MT enters sleep, other MTs can share the channel with fewer MTs (ideally no other MTs). Thus, MTs only spend a very small amount of time in the active mode to prefetch enough data, and then saves power.

B. PBS in Detail

The PBS service model has two parts: a scheduler at the BS side and a proxy at the MT side. The scheduler is used to control the channel access among multiple MTs. The proxy is used to coordinate with the scheduler at the MT side. We describe the algorithm used by the scheduler, and show how the proxy works. Then, we present solutions to deal with channel errors, calculate the computation complexity and prove some properties of the service model.

1) The PBS Scheduler: In PBS, each packet of a flow is assigned a deadline. The scheduler orders the transmission of packets according to their deadlines. If a flow's aggregated service goes beyond the minimum service required to

[1]In this case, in order to provide QoS, the WNI has to stay in active and the scheduler serves the flow in a work-conserving manner.

maintain the QoS[2], it will be removed from the scheduling region until it needs more data to maintain the QoS. As a result, the flow is suspended periodically and the data are transmitted in the form of several runs of packets.

Service accounting: Without loss of generality, for each flow, we assume that the first served packet after entering the scheduling region is indexed by 1. Each packet is assigned a deadline according to the packet length and the flow's data rate. Formally, the deadlines (in seconds) are computed as follows:

$$\begin{aligned} d_i^1 &= e_i \\ d_i^j &= d_i^{j-1} + \frac{l_i^{j-1}}{r_i} \end{aligned} \qquad (1)$$

where p_i^j is the j^{th} packet of flow f_i, e_i is the eligible time of f_i, d_i^j is the deadline of packet p_i^j, l_i^j is the length of p_i^j and r_i is the data rate of f_i (in bps). Suppose, p_i^n is the head-of-line packet of f_i at time t, the ahead-service (in seconds) of f_i, denoted by $ahead_i$, is computed as follows:

$$ahead_i(t) = max(d_i^n - t, 0) + I(i) * \frac{l_i^n}{r_i} \qquad (2)$$

where I(i) returns 1 if f_i is being served, otherwise, it returns 0. Since $ahead_i$ is used to trace the amount of prefetched data at the MT side, it cannot be negative at any time.

Scheduling state management: At the BS side, each flow has two scheduling states: *idle* and *active*. The transitions between scheduling states of f_i (denoted by $state_i$) are controlled by the scheduler. For the purpose of flow control, there is an upper limit of ahead-service for each flow f_i, denoted by $Maxserv_i$. When $ahead_i > Maxserv_i$, the scheduler stops serving it and changes $state_i$ to idle. Suppose the scheduler provides enough ahead-service to f_i; i.e., $ahead_i \geq \phi$, ϕ is a system parameter to represent the lower bound for ahead-service. If there exists another flow, say f_j, which have not got enough ahead-service (i.e.,$ahead_j < \phi$), the scheduler lets f_i yield the channel to other flows by changing $state_i$ to be idle. Whenever $state_i$ is changed to idle, the eligible time of f_i is set to be the current time plus $ahead_i$. At the same time, the scheduler updates the value of the eligible time to indicate when f_i will be moved back in scheduling region again. Whenever f_i is changed to idle, the scheduler will not serve it until the eligible time expires. After the eligible time expires, its state will be changed to active again (not shown in Figure 4).

When $state_i$ is set to idle, the scheduler should notify MT_i which can shutdown its WNI. We assume that the BS can mark the packet transmitted to the MT that will power off its WNI after receiving the marked packet. Since the MT's address is equal to the destination address of the packet, the BS can simply use one bit in the packet header to represent whether the packet is marked or not. When the MT receives the marked packet, it replies an ACK and puts its WNI into sleep. When the BS receives the ACK, it knows that the WNI has been powered off, and suspends the related flow.

[2]At this time, the MT of the flow has enough prefetched data. For simplicity, we say the flow has got enough *ahead-service*.

Notations:
$\mathcal{A}$**:** the set of flows in active state
D_i**:** the deadline of the head-of-line packet of f_i
t: the current time

schedule()
```
1  begin:
2  if ($\mathcal{A} == NULL$)
3      { idle in the time slot; goto begin; }
4  if (no primary flow)
5     select the primary flow $f_i$ according to Eq. (3)
6  if ( $t \geq \arg\min_{j \in \mathcal{A}}\{D_j\} \wedge j \neq i$ )
7       $i = j$; /* the deadline of the secondary flow $f_j$
        will be violated, so serve $f_j$ */
8  $p$=$f_i$.deque(); /* get the packet to be transmitted */
9  if ($ahead_i > Maxserv_i \vee (ahead_i \geq \phi \wedge$
        $\exists j(f_j \in \mathcal{A} \wedge ahead_j < \phi))$
10       mark($p$);
11   send($p$);
12   if (the transmission is successful)
13       { $D_i = D_i + p.length/r_i$;
14         if ($p$ is marked)
15             {$e_i = t + ahead_i; state_i = idle$; } }
16   goto begin
```

Fig. 4. The PBS Algorithm

The scheduler: The PBS scheduler works as follows. When n ($n \geq 1$) flows are active, the scheduler selects one flow as the *primary flow*, and the other $n-1$ flows are *secondary flows*. At any time, the scheduler exclusively serves the primary flow provided that the deadlines of the secondary flows will not be violated. If the deadline of the secondary will be violated, the scheduler has to serve the secondary flow in order to meet the QoS requirement of the flow. In other words, the scheduler serves the primary flow in a work-conserving manner, whereas each secondary flow is served in a non-work-conserving way. As $\frac{(\phi - ahead_i)r_i}{C - \sum_{j \in \mathcal{A}} r_j}$ (See Property 2 in Section III-C) is an approximation of how fast a flow f_i can have enough ahead-service as the primary flow, the scheduler always selects the flow that can take the shortest time to get enough ahead-service as the primary flow. Formally, at time t, the primary flow (f_{prim}) is selected as follows:

$$f_{prim} = \arg\min_{i \in \mathcal{A}}\{\frac{(\phi - ahead_i(t))r_i}{C - \sum_{j \in \mathcal{A}} r_j}\} \qquad (3)$$

where $\mathcal{A}$ is the set of flows in active state and C is the channel capacity. The principle behind Eq (3) is similar to the shortest job first policy, which can minimize the average waiting time. Thus, the average time for each flow to get enough ahead-service is also minimized under PBS.

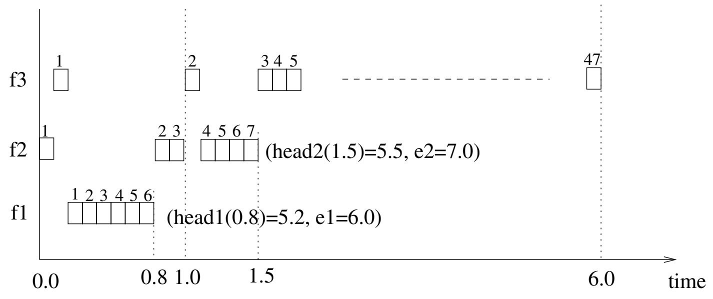


Fig. 5. An illustration of the PBS scheme

Figure 5 shows how the PBS scheme works. There are three backlogged flows (f_1, f_2, f_3) in the system. Each flow has 1Kbps data rate and unlimited $Maxserv$. Suppose $\phi = 5.0$ seconds, $C = 10$ Kbps, and all packets have the same packet length of 1 Kb. At time 0.0, the eligible time of all flows ($f_1 ... f_3$) expires, and the ahead-service of each flow is 0. Suppose f_1 is selected as the primary flow at time 0.0. To meet the deadlines of f_2 and f_3, the scheduler serves p_2^1 and p_3^1 first. From time 0.2 to 0.8, without violating the deadlines of f_2 and f_3, which are equal to 1.0, $p_1^1 ... p_1^6$ are served back-to-back. At time 0.8, according to Eq (2), $ahead_1$ is $d_1^6 - 0.8 = 5.2$, which is greater than ϕ. Thus, f_1 is suspended and its eligible time is set to 6.0. At time 0.8, f_2 becomes the primary flow. Follow the same procedure, f_2 is suspended from time 1.5 to 7.0. After time 1.5, f_3 is the only active flow, and the scheduler serves f_3 until time 6.0 when the eligible time of f_1 expires.

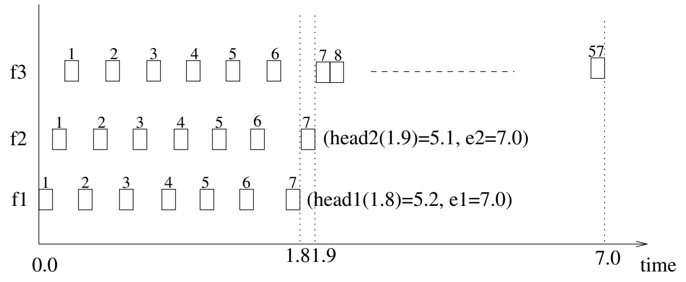


Fig. 6. An illustration of the WFQ scheme

To demonstrate the power efficiency of PBS, we compare the time period of each flow to get enough ahead-service under WFQ [17] and PBS. As shown in Figure 6, under WFQ, f_1 needs 1.8 seconds to get enough ahead-service, and f_2 needs 1.9 seconds. Comparing to the correspondent time periods in Figure 5 (e.g., f_1 needs 0.8 second), we can see that, on average, the time period of each flow to get enough ahead-service under PBS is much smaller than that under WFQ.

Application-aware extensions: Compared with streaming applications, other applications (e.g. FTP, WWW) may not have stringent delay requirements. Suppose a non-streaming flow, say f_k, requires data and the channel utilization is high. If f_k is admitted into the scheduling region, from Eq (3), we can see that the time spent by the primary flow to get enough ahead-service will be increased, and the secondary flows have to wait longer before being the primary flow. Thus, all active flows spend more time in the active state, and the power consumption is also increased. Since f_k does not have strict delay requirement, it is better to postpone the serving time of f_k for a specified time period, denoted by $yield_k$. We assign an integer *relax factor* (σ_i) to each flow f_i, which is bounded by σ_i^{max}. For the flow with strict delay requirement, the upper bound is simply set to 0. For f_i with $\sigma_i > 0$, when its deadline expires, the scheduler decides whether to serve it or not according to the current system utilization, which can be measured by the number of active flows. If the current system utilization is greater than a threshold, denoted by μ_{thresh}, the scheduler lets f_i yield the channel for a period of $yield_i$. Otherwise, f_i is served.

To avoid starvation, an adaptive scheme is used to manage σ_j as follows.
1) When the scheduler decides to let f_j yield the channel, σ_j is decreased by one;
2) When f_j leaves the channel with the ahead-service greater than $\phi + yield_j$, σ_j is increased by one. The service loss due to yielding is compensated by decreasing the ahead-service by $yield_i$.
By using the adaptive scheme, the maximum delay for f_i to be admitted into the scheduling region is $\sigma_i * yield_i$.

2) The PBS Proxy: A proxy is associated with the receiver of each flow. The proxy downloads data from the BS, monitors the amount of prefetched data, and manages the operation modes of the WNI. The proxy coordinates with the server and decides whether the WNI should enter sleep. Similar to the scheduler, the proxy can calculate the ahead-service of each flow according to the flow's data rate, packet length and the arrival time of each packet. If the proxy finds that the packet is marked, the WNI will be shutdown for a time period equal to the calculated ahead-service. In this way, the control overhead can be reduced since the scheduler does not need to tell the length of the sleep period.

If multiple applications are supported, the WNI is shut down only when *all proxies* running on the MT have requested to do so. However, the WNI wakes up if *any proxy* needs it at any time. Since the MT knows all proxies running on it, this can be easily implemented.

3) Dealing with Channel Errors: The wireless communication channel is error prone, and the error is location-dependent and bursty. If channel errors exist, the probability of a successful transmission becomes very low, and hence, bandwidth and power may be wasted during re-transmissions. Similar to [2], [13], [14], we deal with channel errors by swapping time slots from flows suffering channel errors to flows which have good channel conditions. However, we focus on minimizing the influence of channel errors on QoS and power consumption of each flow under the PBS service model. In our approach, every packet is delivered in a DATA-ACK order. If the BS did not receive the ACK of a packet, it knows that the transmission fails. After a transmission failure, the scheduler re-transmits the packet up to three times. If the packet still cannot be delivered, the BS assumes that the flow, say f_i, is experiencing a channel error. In this case, if $ahead_i \geq \phi$, the BS stops f_i. At the same time, the proxy of f_i also realizes the error problem from the transmission failures and shuts down the WNI. If $ahead_i < \phi$, it may not worth to power off the WNI since the actual sleep time could be too short. Thus, the scheduler stops serving f_i for a pre-specified short period, which is called *backoff period*, and the proxy of f_i lets the WNI stay in active. After the backoff period, the scheduler will resume serving f_i. If f_i still suffers from channel errors, the same backoff procedure will be applied to f_i again.

Since most channel errors are bursty, after leaving the channel for some time, the flow may get good channel state when being served. After a flow leaves the channel, the total number of flows sharing the channel decreases and the allocated data rate of each remaining flow increases. As a result, these active MTs spend less time in active, and then reduce the power consumption. Under rare situations, the channel error may last long enough to violate the QoS requirement. At this time, the only solution left is to change the modulation and coding schemes. Since this is not the major concern of this paper, we will not further discuss it.

4) Computation complexity of PBS: It only takes one division and one addition to get the ahead-service (in seconds) at the MT side. At the BS side, the operation consists of the selection of the primary flow at the cost of $O(log(n))$, and the updates for ahead-service per-flow at the cost of $O(n)$. Thus, PBS is computationally feasible in many wireless networks that have a moderate number of flows per BS. MTs can easily get the data rate of the flow, ϕ, Maxserv$_i$, the backoff period and $yield_i$ (for non-streaming applications) during session initialization. Since the performance (in terms of QoS and power efficiency) of PBS is not sensitive to the selection of ϕ even when the system is heavily loaded (see Section IV-D), it does not require precise information about the length of the state transition delay. The only requirement is that ϕ is much larger than the delay.

C. Property of PBS

In this section, we present some important properties of the PBS scheme. We prove that PBS can achieve power efficiency and QoS provision for all flows in the system. For simplicity, we assume that the channel, with a channel capacity of C, is error-free, and each flow has the same data rate r. The proofs are shown in the Appendix.

Property 1: (Delay Guarantee) If Q is the set of backlogged flows in the system, and $|Q|r < C$, the PBS scheduler will serve each packet p_i^j according to:

$$s_i^j - d_i^j \leq (|Q| - 1)\frac{L^{max}}{C} \tag{4}$$

where s_i^j is the time when the server starts serving p_i^j and L^{max} is the maximum packet length.

Property 2: (Power Efficiency) Suppose Q is the set of backlogged flows in the system and $|Q|r < C$. Consider the process that each flow prefetches the ahead-service greater than or equal to ϕ and leaves the channel. If $\phi r \gg L_{max}$, the average active time under PBS and under weighted fair queuing (WFQ), denoted by $\bar{T}_{act,PBS}$ and $\bar{T}_{act,WFQ}$ respectively, follows:

$$\frac{\bar{T}_{act,WFQ}}{\bar{T}_{act,PBS}} > \frac{\frac{\phi r - L^{max}}{C/|Q| - r}}{\sum_{i=1}^{|Q|} \frac{(|Q|-i+1)(\phi r + (|Q|-i+1)L^{max})}{|Q|(C-(|Q|-i+1)r)}} \tag{5}$$

Property 2 gives the *lower bound* ratio of the average active time under WFQ to that under PBS. We give an example to show some numerical results of the ratio as a function of $|Q|$. Suppose $C = 400Kbps$, $r = 50Kbps$, $\phi = 500ms$, and $L^{max} = 1000bits$. As shown in Figure 7, when $|Q|$ increases from 2 to 6, the ratio lower bound increases from 1.26 to 2.14. This shows that PBS is more power efficient than WFQ.

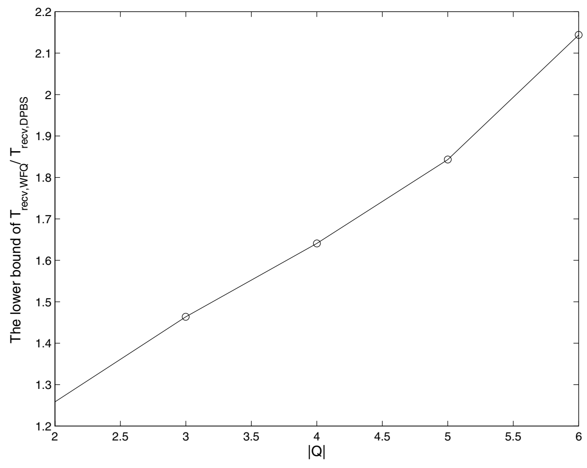
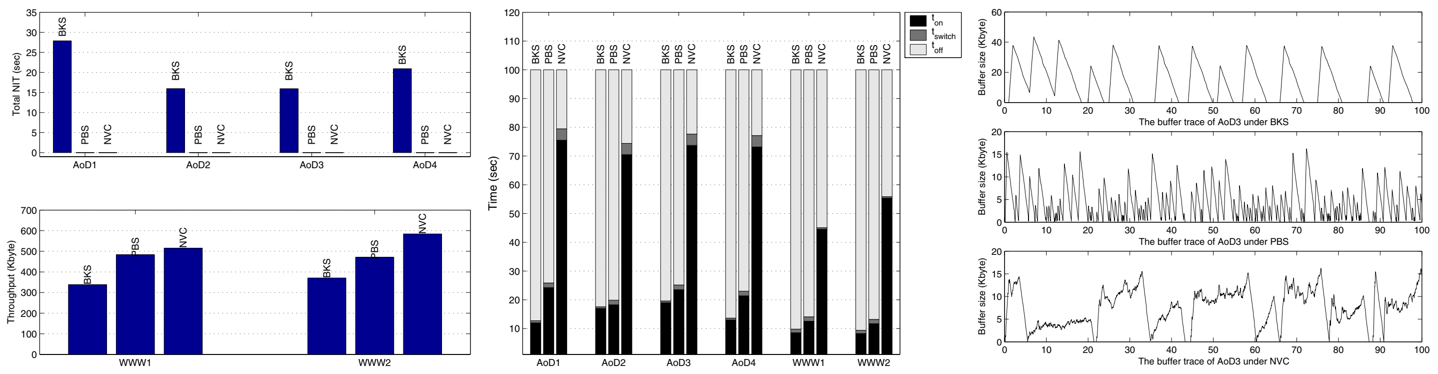


Fig. 7. The numerical results of the lower bound of $\frac{\bar{T}_{act,WFQ}}{\bar{T}_{act,PBS}}$

IV. PERFORMANCE EVALUATIONS

A. The Experimental Setup

We evaluate the performance of the PBS service model through case studies consisting of *Audio-on-Demand (AoD)* and WWW services. The AoD streaming service is evaluated through trace-driven simulation. We downloaded a demo MP3 audio streams from [16] as the trace source. Since the MP3 streams are based on variable bit rate (VBR), we extracted information of every frame, i.e. the frame size, the frame bit rate, and the frame sample rate, to get the bit rate of each packet. There is an interval between successive songs, which is distributed randomly between $[0.0, 2.0]$. The WWW service



(a) Quality of service (b) Power consumption (c) Buffer trace of AoD3

Fig. 8. Performance comparisons among BKS, PBS, and NVC without channel errors

is emulated by a simple ON/OFF traffic model that mimics the web user access behavior. An ON period will start when a new Web page is requested. Once the run of data has finished, there is an OFF period during which the user studies the information just downloaded. We assume the total page size of an ON period is exponentially distributed with the mean of 12 KB, and the length of an OFF period is exponentially distributed with the mean of 2.0 seconds. The capacity of the wireless channel is assumed to be 384 Kbps, and is based on TDMA. Each time slot is $2.5ms$, which can be used to transmit 112 Bytes of data (not including the header). Each MT represents a user having an AoD or WWW flow. Each flow is allocated a data rate of 56 Kbps. The simulation time is 100 seconds. For the WNI, we assume $T_{on\rightarrow off} = 5ms$ and $T_{off\rightarrow on} = 10ms$. The PBS server sets $\phi = 80ms$ and $\mu_{thresh} = 50\%$. In addition, $yield_i$ and backoff period of flow f_i are set to be 100 ms and 15 ms respectively. The relax factor is set to be 2 for WWW flows, and the $Maxserv$ is set to be $2000ms$ for AoD flows.

Channel errors are modeled by a two state Markov chain, which has two states: good and bad. The probability from a good state to a bad state is 0.05, and from a bad state to a good state is 0.08. When the channel is in good state, packets are transmitted correctly. When the channel is in bad state, packet transmissions will fail.

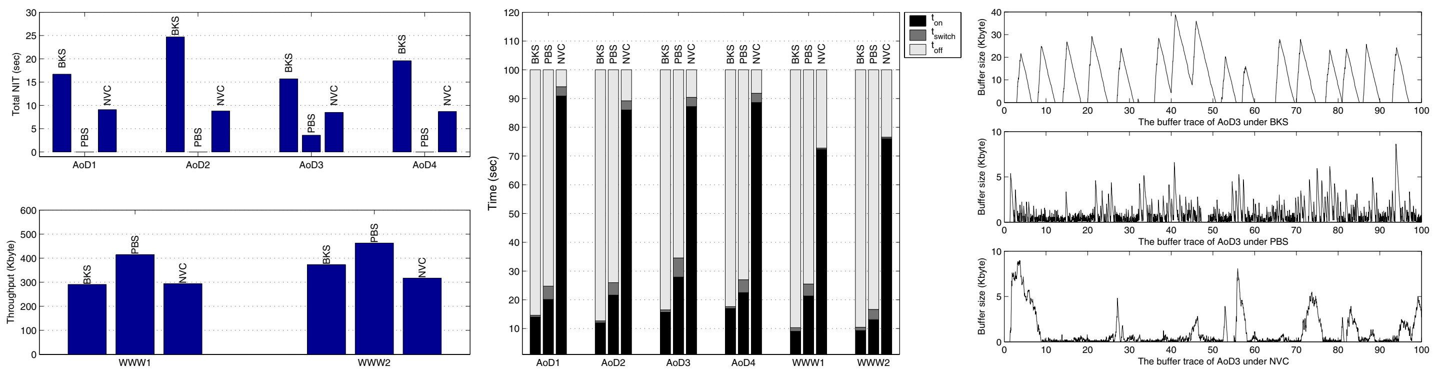
We evaluate the performance of the proposed service model based on the following factors: the amount of prefetched data, the QoS and the time spent in active, sleep and state transition. We use the total *noticeable interrupt time* (NIT) to measure the quality of the playback audio. Since tiny interrupts are not noticeable by human being, only continuous interrupts which are greater than $20ms$ are counted as NITs. This is less than the normal delay requirement for telephone voice service since we are dealing with music here. In the simulations, the total NIT is equal to the aggregated noticeable interrupt time intervals. Since WWW service does not have strict delay requirement, its QoS is measured by the *throughput*, which is equal to the total amount of data (in bytes) transmitted. To evaluate the time spent in each operation mode, t_{on}, t_{off}, t_{switch} are used to denote the total time spent in active, sleep and state transition respectively.

We compare the performance of the PBS scheduling with the bulk scheduling (BKS) and the Non-work-conserving Virtual Clock (NVC) scheduling scheme. We choose the bulk slot time to be 1.0 second for BKS. There is a buffer on the MT side in the BKS approach and the NVC approach. In BKS, the MT stays in sleep after being served until the prefetched data run out. In NVC, the MT with an AoD flow goes to sleep when the buffer is full and wakes up when the buffer is near empty. The evaluation considers two scenarios. In Scenario 1, the channel is error free. In Scenario 2, an error-prone channel is considered. Finally, we evaluate the impacts of ϕ.

B. Scenario 1: Error-free Channel

In this scenario, we show the fundamental difference among PBS, BKS, and NVC. The evaluation includes four AoD flows and two WWW flows. From Figure 8 (a), we can see that the playback quality of the PBS approach and the NVC approach are perfect since their total NITs are 0. Compared with PBS and NVC, the BKS approach provides poor QoS since the total NIT of each AoD flow is greater than 15 seconds, and the throughput of each WWW flow is much less than that under PBS and NVC. Since there are six flows sharing the channel, it is highly possible that more than one flow request data at the beginning of a bulk slot. The BKS scheduler only randomly selects one winning flow to serve. As a result, other losing flows cannot be served during the bulk slot, and their delay requirements are violated.

As shown in Figure 8 (a), flows WWW1 and WWW2 have higher throughput in NVC than PBS, since PBS lets WWW1 and WWW2 yield the channel when the system's utilization is greater than 50%. On the other hand, as shown in Figure 8 (b), the power consumption of each WNI under NVC is significantly higher than that under PBS and BKS. Since the data rate is quite high (i.e., 56 Kbps on average), many of the inter-packet arrival periods are less than $T_{on\rightarrow off} + T_{off\rightarrow on}$. As a result, the WNI cannot always go into sleep in the NVC approach. Even though the WNI can enter sleep, the actual sleep period is reduced by $T_{on\rightarrow off} + T_{off\rightarrow on}$.



(a) Quality of service (b) Power consumption (c) Buffer trace of AoD3

Fig. 9. Performance comparisons among BKS, PBS, and NVC with channel errors

Figure 8 (c) shows how these three approaches work through the buffer trace of flow AoD3. When the buffer size goes up, the WNI receives data in the active mode. During the time period when the buffer size goes down until the buffer size is near zero, the WNI stays in sleep. The slope of the buffer increment indicates how fast the flow fills the buffer. For BKS, since the flow is always served alone during a bulk slot, its actual data rate is equal to the channel capacity, which is the ideal power efficiency. From Figure 8 (c), we can see that the slope of the buffer increment under PBS is almost equal to that under BKS, which shows that the power efficiency under PBS is good. Under PBS, at any time, the scheduler serves the primary flow alone provided that the deadlines of the secondary flows will not be violated. As a result, the actual data rate of the primary flow may be quite high when there are few secondary flows. On the other hand, by suspending the primary flow that gets enough ahead-service, the new selected primary flow competes the channel with less flows and gets higher data rate.

Compared with PBS and BKS, the slope under NVC is much less than that under BKS during some time period; e.g., during the time period of $(53.0, 57.0)$. Since the data rate of each flow is quite high, the receiver's WNI has very few chances to go to sleep under NVC. At this time, the scheduler serves the flows in a work-conserving manner. Since a large number of flows share the channel, the actual data rate of each flow is small, which reduces the power efficiency.

From this example, we can see that PBS combines the advantages of BKS and NVC. Even though the throughput of WWW services under PBS is not as good as that under NVC, considering the saved power, we believe the benefit outweighs the disadvantages.

C. Scenario 2: Error-Prone Channel

In this subsection, we evaluate the impact of channel errors on three scheduling approaches. We introduce channel errors in this scenario and assume that the channel errors are bursty and location-dependent. We assume only AoD3 and WWW1 experience channel errors, while other users still have error-free channels.

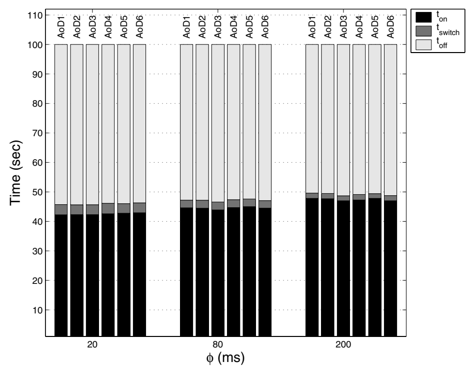
As shown in Figure 9 (a), when channel errors exist, the playback quality of the BKS approach and the NVC approach is much worse than the PBS approach. The poor playback quality of the BKS approach has been explained in Scenario 1, and the reason is still valid for this example. Since the NVC approach does not consider how to deal with bursty and location-dependent channel errors, flows that have good channel conditions are affected by the two flows with channel errors. As a result, the playback quality of each flow is severely degraded. In contrast, during each time slot, the PBS scheduler lets the flow suffering from channel errors yield the channel to other flows with clean channel, so that flows with clean channel can still have good QoS.

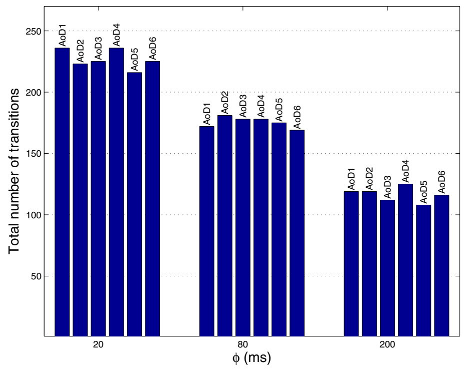
As for AoD3, which suffers from channel errors, its playback quality is not as good as AoD1 and AoD2 which don't have channel errors, but it is still much better than that under BKS and NVC. In the PBS approach, when AoD3 has good channel, it prefetches data. With the prefetched data, when the channel condition is bad, AoD3 can leave the channel for a while. Since channel errors are bursty, this mechanism can offset the impact of channel errors. However, if channel errors last for a long time, the QoS may be violated; e.g., total NIT of AoD3 becomes 3.6 seconds.

As shown in Figure 9 (b), t_{on} and t_{switch} of each flow under PBS is greater than that in scenario 1, since the flows with channel errors are more likely to stay in the scheduling region due to lack of enough ahead-service. According to the state management scheme of PBS, other flows are suspended when their ahead-service is equal to or a little bit more than ϕ. Since ϕ is $80ms$, the receiver's WNI of the flows cannot sleep for a long time, and the number of state transitions increases. This also explains why t_{switch} increases for flows with channel errors, and can be verified by the buffer trace of AoD3 in Figure 9 (c). As we can see, under PBS, the average amount of data in the buffer is less than that in Figure 8 (c).

D. The Impacts of ϕ

As ϕ increases, the number of transitions between sleep and active drops. However, if ϕ is too large, the time spent in getting enough ahead-service will be long. When the workload


(a) Power consumption


(b) Total number of transitions

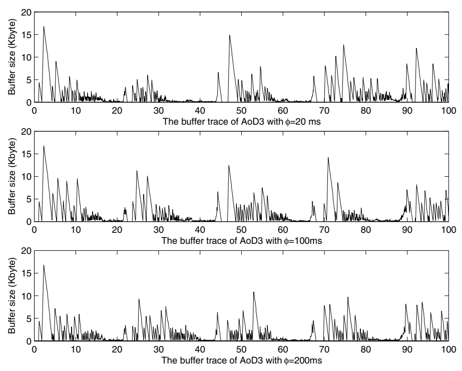

(c) Buffer trace of AoD3

Fig. 10. The impacts of ϕ on PBS

of the system is not heavy, in most cases, the number of flows with insufficient ahead-service is small. As a result, the actual data rate of the flows is high and then the time spent in getting enough ahead-service is very short. Thus, the PBS is not sensitive to ϕ when the workload is small. Since the average rate of each AoD flow is 56Kbps, the maximum number of AoD flows can reach six if the average flow rate is used for admission control. In this scenario, we increase the workload to six AoD flows, and we assume that the channel is error free. We evaluate the performance when ϕ is $20ms$, $80ms$ and $200ms$ respectively. All the NITs of the flows are 0, and we do not show them due to the limited space.

The simulation results are shown in Figure 10. When ϕ increases, the time to get enough ahead-service increases and t_{on} becomes longer. As ϕ becomes smaller, the number of state transitions increases, and t_{switch} becomes larger. However, the sum of the active time and the state transition time under different values of ϕ does not have too much difference. If the power consumed during state transition is similar to that in the active state, the performance of PBS is not sensitive to ϕ. However, if the power consumption of the state transition is much higher, ϕ cannot be too small.

Since the total transition time is very close for different ϕ in Figure (10) (a), we use Figure (10) (b) to show the number of transitions for different ϕ. As can be seen, the number of transitions when $\phi = 200ms$ is about half of that when $\phi = 20ms$. When ϕ decreases, for a given number of flows, the time spent in getting ahead-service increases as ϕ decreases. Thus, each flow takes a small amount of time to receive data. However, a small ϕ prevents the WNI from staying sleep for a long time period, which makes the WNI switch from on to off, and off to on more frequently. As a result, t_{switch} increases. Due to similar reason, t_{switch} decreases as ϕ increases. From Figure 10 (c), we can also see that the behavior of the WNI is different with different ϕ.

V. Related Work

Power management in wireless networks has received considerable attention. IEEE 802.11 [8] supports power saving mode in which the WNI only needs to be active periodically. In a wireless LAN, the WNI in sleep mode only wakes up periodically to check for possible incoming packets from the BS. The BS transmits a *beacon frame* after a regular *beacon interval*. In each beacon frame, a *traffic indication map* (TIM) contains information about which WNI has buffered incoming packets. If the WNI finds that it has incoming packets, it should stay active to receive the packets. However, this mechanism does not work well for streaming applications since the continuously arriving packets forces TIMs to always report new data, keeping the WNI stay active.

An energy conserving medium access control (EC-MAC) scheme for wireless and mobile ATM networks was proposed in [4]. EC-MAC was designed for supporting multimedia traffic and providing QoS for wireless ATM networks. The authors proposed a priority frame-based round robin scheduling scheme considering dynamic reservation update and error compensation. Power saving is achieved by allocating contiguous time slots for each flow. Therefore, in each frame, the WNI only needs to be active during its data phase. However, EC-MAC does not consider the state transition delay. If the frame length is too short, the WNI may not be able to go to sleep due to the transition delay. To achieve high power efficiency, the frame length should be significantly increased, which may increase the queuing delay. Compared with EC-MAC, the RBS scheme considers the issue of state transition delay, and allocates bandwidth on the granularity of per packet rather than per frame, which is flexible to handle the problems of variable packet lengths and dynamic packet arrival patterns.

Prabhakar *et al* [19] studied power conservation with regard to scheduling. They show that the power consumption can be significantly reduced by lowing the transmission power and transmitting the packet over a long period of time. Based on this motivation, the Lazy Packet Scheduling (LPS) approach is proposed to reduce the transmission rate for every packet without violating the deadline of each packet. The approach has been proven to be power optimal. The main difference between LPS and RBS is that: LPS focuses on reducing the power consumed by the WNI of the sender by changing the transmission rate, whereas RBS focuses on reducing the power consumption of the WNI of the receiver (i.e. the MTs) by powering off the WNI.

In [11], a transport layer protocol was proposed to save power. The protocol coordinates the sender and the receiver to be active or idle. The state transition is based on the state of the sender's output buffer. When there is no data to send for a specified period, the sender will notify the receiver to go to sleep for a while. The idea of our approach is along the same direction. Different from their work, we also consider QoS provision.

Fitzek and Reisslein [6] proposed a prefetching protocol to support high performance streaming applications over wireless links. Parts of the ongoing media streams are prefetched into buffers in clients according to a *join-the-shortest-queue* (JSQ) policy. The JSQ dynamically allocates more bandwidth to the clients with small buffered data while allocating less bandwidth to the clients with large prefetched reserves. With the buffered data, the clients can have smooth playback quality during the periods of adverse transmission conditions on the wireless links. The basic idea of PBS is similar to JSQ. However, PBS focuses on the aspect of power conservation of WNIs, while JSQ deals with channel errors of wireless links. Furthermore, PBS also considers how to achieve power efficient communication with regard to scheduling.

VI. Conclusions

In this paper, we presented a deadline-based priority bulk scheduling (PBS) service model to save power and provide QoS guarantees. The PBS scheduler decides to serve or suspend a flow based on the amount of prefetched data. Among the flows with insufficient prefetched data, one flow is selected as the primary flow and the others are secondary flows. The primary flow is exclusively served provided that the deadlines of the secondary flows will not be violated. The flow with enough buffered data will be suspended until the buffered data run out. As some flows do not have strict delay requirement, the PBS scheduler may let these flow yield the channel for a while if the workload of the system is high. With prefetched data, the WNI can sleep long enough to offset the impact of the state transition delay. By suspending some flows, other active flows sharing the channel can obtain more bandwidth and take less time to fill the buffer. Analysis has been given to prove that the service model has delay guarantee and is more power efficient than any other rate-based fair queuing service model. Extensive experiments were carried out to evaluate the proposed methodology. Compared to NVC and BKS, the PBS service model can significantly reduce the power consumption and provide excellent QoS.

As future work, we will extend the PBS service model to further improve its error-resiliency. We will add channel condition into our model when determining the priority of the flow, and adjust the wakeup time based on the channel condition of the flow.

References

[1] J. Bennett and H. Zhang, "WF2Q: Worst-case fair weighted fair queueing," *INFOCOM'96*, March 1996.

[2] P. Bhagwat, P. Bhattacharya, A. Krishma and S. Tripathi, "Enhancing throughput over wireless LANs using channel state dependent packet scheduling," *INFOCOM'96*, March 1996.

[3] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "An Energy-Efficient Coordination Algorithm for Topology Maintaince in Ad Hoc Wireless Networks," *ACM Mobicom*, 2001.

[4] J. C. Chen, K. M. Sivalingam, P. Agrawal and R. Acharya, "Scheduling Multimedia Services in A Low-Power MAC for Wireless and Mobile ATM Networks," *IEEE/ACM Transactions on Multimedia*, vol. 1, no. 2, June 1999.

[5] Reuters company news, "Music via Mobiles Hits High note with Young Crowd," *http://biz.yahoo.com/rc/020611/column_pluggedin_1.html*, 2002.

[6] F. H. Fitzek and M. Reisslein, "A Prefetching Protocol for Continuous Media Streaming in Wireless Environments," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, October 2001.

[7] P. Goyal, H. Vin, and H. Cheng, "Start-Time Fair Queuing: A Scheduling Algorithm for Integrated Serv ices Packet Switching Networks," *ACM SIGCOMM'96*, August 1996.

[8] IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Spec," *IEEE 802.11 standard*, 1999.

[9] E. Jung and N. Vaidya, "A power control MAC protocol for Ad Hoc Networks," *ACM Mobicom*, 2002.

[10] R. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless Web access with bounded shutdown," *ACM Mobicom*, 2002.

[11] R. Kravets and P. Krishnan, "Power Management Techniques for Mobile Communication," *MobiCOM'98*, October 1998.

[12] Q. Li, J. Aslam and D. Rus, "Online Power-aware Routing in Wireless Ad-hoc Networks," *MobiCOM'01*, July 2001.

[13] S. Lu, T. Nandagopal and V. Bharghavan, "A Wireless Fair Service Algorithm For Packet Cellular Networks," *MobiCOM'98*, October 1998.

[14] T.S.E. Ng, I. Stoica and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," *INFOCOM'98*, March 1998.

[15] Nokia, "Nokia 5510 Specifications," *http://www.nokia.com/phones/5510/specifications.html.*

[16] MPEG Org, "MP3 test streams," *http://www.mpeg.org/MPEG/mp3.html.*

[17] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: single node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, June 1993.

[18] R. Powers, "Batteries for Low Power Electronics," *Proc. IEEE*, vol. 83, no. 4, pp. 687–693, April 1995.

[19] B. Prabhakar, E. Uysal-Biyikoglu, and A. El Gamal, "Energy-Efficient Transmission over a Wireless Link via Lazy Packet Scheduling," *IEEE INFOCOM'01*, March 2001.

[20] T. Simunic and S. Boyd, "Managing power consumption in networks on chips," *ACM DATE*, 2002.

[21] S. Sinha and C.S. Raghavendra, "PAMAS: Power Aware Multiple Access Protocol with Signaling for Ad Hoc Networks," *Computing Communication Review*, vol. 28, no. 3, pp. 5–26, 1998.

[22] M. Stemm and R. H. Katz, "Measuring and Reducing Energy Consumption of Network Interfaces in Handheld Devices," *IEICE Transactions on Communications*, vol. E80-B, no. 8, August 1997.

[23] J. Xu and R. J. Lipton, "On fundamental tradeoff between delay bounds and computational comp lexity in packet scheduling algorithm," *ACM SIGCOMM'02*, August 2002.

[24] Y. Xu, J. Heidemann and D. Estrin, "Geography-informed Energy Conservation for Ad Hoc Routing," *Mobicom'01*, July 2001.

[25] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Sw itching Networks," *Proceedings of the IEEE*, vol. 83, no. 10, October 1995.

[26] H. Zhu, G. Cao, G. Kesidis and C. Das, "An Adaptive Power-Conserving Service Discipline for Bluetooth ," *IEEE ICC'02*, 2002.

Appendix

Proof of Property 1:

Proof: The set Q can be partitioned into two subsets: the set of active flows, denoted by $\mathcal{A}$, and the set of idle flows denoted by $\mathcal{I}$. At any time t, flow f_i has two cases:

- **case 1:** $f_i \in \mathcal{I}$. Suppose p_i^j is the last packet served before f_i was suspended. Since $d_i^j > t$ (otherwise $f_i \in \mathcal{A}$) and $s_i^j \leq t$ (since f_i was suspended before t), the property holds.
- **case 2:** $f_i \in \mathcal{A}$. Suppose the head-of-line packet of f_i is p_i^j at time t and $d_i^j = t$. Since the definition of the deadline of each packet is the same as the *start-time first fair queuing* (SFQ), and f_i has the highest priority under PBS provided that $d_i^j \leq t$, with similar arguments to Theorem 4 of [7] and $\mathcal{A} \subseteq Q$, we get:

$$\begin{aligned} s_i^j - d_i^j &\leq \sum_{k \in \mathcal{A} \wedge k \neq i} \frac{L^{max}}{C} \\ &\leq \frac{(|Q|-1)L^{max}}{C} \end{aligned}$$

□

Proof of Property 2:

Proof: If each flow is served under *generalized processor sharing* (GPS) [17], the actual data rate of each flow is equal to $\frac{C}{|Q|}$. Then, flow f_i's aggregated service (in bits) during a time period of Δt is equal to $\frac{C}{|Q|}\Delta t$. Now, let's consider the situation that the scheduler applies the WFQ scheme and each flow is served with the granularity of packet.

Since all the flows are continuously backlogged and have the same data rate, the WFQ scheduler behaves similar to the worst-case fair weighted fair queuing (WF^{2}Q) [1] scheduler. Following Theorem 1 of [1], the relationship between the amount of f_i's aggregated service under GPS and WFQ during the interval Δt, denoted by $W_{i,WFQ}(\Delta t)$ and $W_{i,GPS}(\Delta t)$ respectively, follows $W_{i,WFQ}(\Delta t) - W_{i,GPS}(\Delta t) < L^{max}$. Hence, we have

$$W_{i,WFQ}(\Delta t) < \frac{C}{|Q|}\Delta t + L^{max} \tag{6}$$

Without loss of generality, suppose flow f_1 is the first flow that leaves the channel, and f_1 spends T_1 to get the ahead-service equal or greater than ϕ. T_1 needs to satisfy $W_{i,WFQ}(T_1) - rT_1 \geq \phi r$. With Ineq (6), we get

$$T_1 > \frac{\phi r - L^{max}}{C/|Q| - r} \tag{7}$$

Since f_1 is the first flow that gets enough ahead-service, according to Ineq (7), we have:

$$\bar{T}_{act,WFQ} > \frac{\phi r - L^{max}}{C/|Q| - r} \tag{8}$$

Under PBS, at any time, set Q can be partitioned into set $\mathcal{A}$ and $\mathcal{I}$. The flows in $\mathcal{I}$ have got enough ahead-service and left the channel, while flows in $\mathcal{A}$ are served. Since the secondary flows in $\mathcal{A}$ are served in non-work-conserving manner, similar to the proof of Lemma 1 in [7], the aggregated service of secondary flow f_j during any time interval Δt, denoted by $W^s_{j,PBS}(\Delta t)$, follows:

$$r\Delta t - L^{max} \leq W^s_{j,WFQ}(\Delta t) \leq r\Delta t + L^{max} \tag{9}$$

Suppose the primary flow f_i takes $\Delta T_{i,PBS}$ to get ahead-service greater than or equal to ϕ. Similar to the proof of Theorem 2 in [7], the aggregated service of f_i during $\Delta T_{i,PBS}$, denoted by $W^p_{i,PBS}(\Delta T_{i,PBS})$, follows:

$$\begin{aligned} &W^p_{i,PBS}(\Delta T_{i,PBS}) \\ &\qquad \geq \quad \Delta T_{i,PBS}(C - |\mathcal{A}|r) - \textstyle\sum_{j \in \mathcal{A} \wedge j \neq i} L^{max} \\ &\qquad \geq \quad \Delta T_{i,PBS}(C - |\mathcal{A}|r) - (|\mathcal{A}|-1)L^{max} \end{aligned} \tag{10}$$

Since f_i leaves the channel whenever it gets the ahead-service greater than or equal to ϕ, $\Delta T_{i,PBS}$ satisfies: $W^p_{i,PBS}(\Delta T_{i,PBS}) - \phi r \leq \phi r + L^{max}$. Combined with Ineq (10), we get

$$\Delta T_{i,PBS} \leq \frac{\phi r + |\mathcal{A}|L^{max}}{C - |\mathcal{A}|r} \tag{11}$$

Without loss of generality, suppose the order of primary flows during the whole process is: $f_1, f_2 ... f_{|Q|}$. Due to $\phi r \gg L^{max}$ and Ineq (9), each flow cannot have enough ahead-service when it is a secondary flow. As a result, the time spent by f_i to leave the system, denoted by $T_{i,PBS}$, follows:

$$T_{i,PBS} = \sum_{k=1}^{i} \Delta T_{k,PBS} \tag{12}$$

During the time period when f_i is the primary flow, $|\mathcal{A}| = |Q| - i + 1$. Hence, with Ineq (11) and Eq (12), we get:

$$\bar{T}_{act,PBS} \leq \sum_{i=1}^{|Q|} \frac{(|Q|-i+1)(\phi r + (|Q|-i+1)L^{max})}{|Q|(C - (|Q|-i+1)r)} \tag{13}$$

With Ineq (8) and Ineq (13), it is easy to find out that the property holds. □

Different fair queuing model may have different fairness property [25]. However, since we assume that each flow is continuously backlogged and has the same date rate, with the results of [23], Ineq (6) still holds for other rate-based fair queuing models. Thus, Property 2 is also valid for other rate-based fair queuing models.