

On Supporting Power-Efficient Streaming Applications in Wireless Environments

Hao Zhu and Guohong Cao

Abstract—Reducing the power consumption of the wireless network interface (WNI) is an effective way to prolong the battery lifetime of the mobile terminal. It takes some time for the WNI to transit from the power-saving mode to the active mode. This transition delay and the error-prone wireless link bring many challenges for designing power-aware and QoS-aware service models. In this paper, we present a novel power-conserving service model for streaming applications over wireless networks. At the base station side, a new scheduling algorithm, called *rate-based bulk scheduling* (RBS), is designed to decide which flow should be served at which time. The mobile terminal relies on a proxy to buffer data so that the WNI can sleep for a long time period to save power. To deal with channel errors, a novel adaptive technique is presented to adjust the sleep time of the WNI according to the channel condition. Through analysis, we prove that RBS can provide delay guarantee and it is more power efficient than other rate-based fair queuing algorithms. We use Audio-on-Demand as a case study to evaluate the performance of RBS. Experimental results show that RBS achieves excellent QoS provision for each flow and significantly reduces the power consumption.

Index Terms—Power-aware, simulations, QoS, scheduling, wireless networks.

1 INTRODUCTION

WITH the rapid development of wireless network technologies, it becomes feasible to support streaming applications such as audio-on-demand and video-on-demand on wireless networks. As predicted by IDC [5], there will be 37 million users listening to music that is delivered through wireless channels by 2006. Since most mobile terminals (cellular phones, laptops, PDAs, etc.) are powered by battery, many researchers are working on techniques to reduce the power consumption of these mobile terminals (MTs). As the wireless network interface (WNI) accounts for a large part of the power consumed by the MT (as much as 70 percent of the total power in Nokia 5510 [18]), it is important to carefully design communication protocols to reduce the power consumption of the WNI.

Understanding the power characteristics of WNIs is important for the efficient design of communication protocols. A typical WNI may exist in *active* or *sleep* mode. There are three active modes: transmit, receive, and idle. Many studies [12], [24], [31] show that the power consumed in any active mode is similar, which is significantly higher than the power consumed in the sleep mode. As a result, most works on power management concentrate on putting the WNI into sleep when it is idle. This principle has been applied to different layers of the network hierarchy. At the MAC layer, protocols [8], [11], [20], [34] are proposed to put the WNI into sleep to reduce the power consumption 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. At the network layer, routing protocols [12], [31] have been proposed to only keep some necessary MTs awake and put others into sleep. Periodically, MTs alternate their roles to prolong the network operating time. Stemm and Katz [24] have studied transport layer approaches and application-driven approaches to help power down the WNI. These schemes mainly focus on reducing the power consumption, which is enough for most data applications without QoS requirements, but may not be enough for streaming applications. As a result, new protocols are needed to *achieve power saving without violating QoS*.

To consider both power conservation and QoS provision, we need to decide when and how long the WNI should be powered off. If the WNI sleeps too long, the application may not have enough data to meet the QoS requirements since no data can be delivered to the MT when the WNI is in sleep. On the contrary, to favor the QoS requirements, the power management may be too conservative to save power. Ideally, the WNI should be turned on exactly when the MT needs the data to satisfy the QoS requirements. However, this may not be true in practice due to the following reasons:

- It takes some time for the WNI to wake up from sleep. As reported in [23], the transition time from active to sleep and back to active is on the order of tens of milliseconds. If the interpacket arrival time of the application is too small, due to the state transition delay, the WNI cannot enter sleep and provide QoS.
- The WNI requires a significant amount of energy to go from sleep to active [29]. Generally, the power consumption increases as the transition delay increases. Even though the delay is very short (e.g., hundreds of microseconds), as stated in [29], energy dissipation by the start-up transition is still on the same order of that in the active mode. In order to save power, the number of state transitions of the

• H. Zhu is with the Department of Electrical and Computer Engineering, Florida International University, 10555 W Flagler Street, EC 2946, Miami, FL, 33174. E-mail: hao.zhu@fiu.edu.

• G. Cao is with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802. E-mail: gcao@cse.psu.edu.

Manuscript received 13 Feb. 2004; revised 2 July 2004; accepted 12 July 2004; published online 27 May 2005.

For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0029-0204.

WNI should be low. Also, the power saving would be very small if the sleeping period is too short.

- Wireless links are error-prone and channel errors are typically location-dependent, time-varying, and bursty [13], [15], [17], [27]. Due to the stringent delay requirement of streaming applications, channel errors may bring significant QoS degradation and power consumption due to retransmissions. Further, the dynamic nature of channel errors raises the problem of when to activate the WNI. Since the flow may still suffer from channel errors when the WNI wakes up, the MT may not be able to receive the data in time and violate QoS requirements.

In this paper, we propose a new scheduling algorithm, called *rate-based bulk scheduling* (RBS), to utilize the client buffer to save power and provide QoS. With RBS, the scheduler decides to serve or suspend a flow based on the amount of buffered data. The flow with insufficient buffered data will be served with rate-based fair queuing for the purpose of QoS provision, whereas the flow with enough buffered data will be suspended for a certain amount of time. With buffered data, the WNI can sleep long enough to offset the impacts of state transition delay and channel errors. By suspending some flows, other active flows sharing the channel can obtain more bandwidth and take less time to fill up the buffer. To tolerate severe channel errors, a novel adaptive scheme is used to control the sleeping time of the WNI based on the channel condition. Through analysis, we prove that RBS can provide delay guarantee and it is more power efficient than other rate-based fair queuing algorithms. We use Audio-on-Demand (AoD) as a case study to evaluate the performance of RBS. Experimental results show that RBS 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 3, we describe the details of RBS. Section 4 evaluates the performance of RBS. Related work will be presented in Section 5. Section 6 concludes the paper.

2 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). The downlink channel can be accessed in a TDMA manner. Location-dependent errors may happen due to channel fading, interference, etc., [13], [15], [17], [27]. Similar to many existing works [13], [14], [15], [17], we assume that the BS has a mechanism to predict the channel condition. Based on the channel condition, like some existing systems (e.g., EDGE [27] and HDR/EV-DO [1]), the BS selects the most suitable modulation and coding scheme to transmit the impending packet. For each time slot, there are a set of possible transmission rates $\{0, C^1, C^2, \dots, C^M\}$. Since the BS selects the transmission rate according to the channel condition, we can

use the present transmission rate as the indicator of the current channel condition. For wireless systems that do not apply rate adaptation to deal with channel errors (e.g. power-adaptive CDMA systems), we have proposed the other solution in [35].

A closed-loop mechanism [26] at each MT could be employed in order to let the BS accurately predict the downlink channel condition. Specifically, the MT measures the received signal-to-noise ratio (SNR) and compares it with the desired SNR. Depending on whether the measured SNR is above or below the desired SNR, the MT instructs the BS to change the modulation and coding scheme through the uplink or a uplink control channel [26]. The impact of the closed-loop mechanism on power efficiency of the MT can be significantly reduced by decreasing the rate of feeding back the channel condition report. For example, when the WNI of the MT is in sleep, the rate can be reduced from the default 800 bps to 50 bps by letting the MT operate at a different closed-loop mode [26]. In addition, with cutting-edge low-power VLSI techniques (e.g., selective circuit power-down [21]), the power consumption of the WNI in the sleep mode can be further reduced. Consequently, even with the closed-loop mechanism, the power consumption of the WNI in sleep is much less than that in active. This claim can be simply verified by the difference between the standby time and the talk time of current cellular phones. For example, the standby time of Motorola V60i is 220 hours whereas its talk time is four hours [16].

Since an MT spends much less power in sleep, the WNI should enter sleep (not idle) as long as possible to save power. As a result, the power saved by different schemes can be measured by the accumulated WNI sleep time. When measuring the WNI sleep time, we need to consider the state transition time between active and sleep. In order to accurately measure the power consumption, we use $T_{off \rightarrow on}$ to denote the time to transit from sleep to active and $T_{on \rightarrow off}$ to denote the time to transit from active to sleep.

3 RATE-BASED BULK SCHEDULING (RBS)

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

3.1 Background and Motivation

In order to provide guaranteed service over a shared link, several rate-based service disciplines have been proposed [33]. The principle of these service models is to provide each flow with a guaranteed data rate without being affected by other misbehaving flows sharing the link. A scheduling algorithm can be classified as *work-conserving* or *nonwork-conserving*. In the work-conserving scheduling, a server is never idle when there is a packet to send. In the nonwork-conserving scheduling, a packet is not served until it is eligible [33], 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 1, 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.

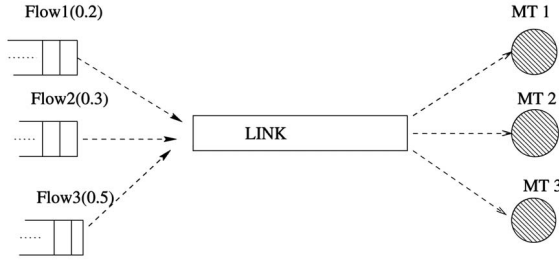


Fig. 1. An example of work-conserving link sharing.

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 Fig. 1, suppose three flows 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 percent of the power has been wasted.

Nonwork-conserving Virtual Clock (NVC): Nonwork-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 *nonwork-conserving* [33] 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 *Nonwork-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 $T_{off \rightarrow on} = 10ms$; if the packet interval time, denoted by T_{intvl} , of the flow is less than $10ms$, we have $T_{on \rightarrow off} + T_{off \rightarrow on} > T_{intvl}$. Thus, the NVC scheme cannot put the WNI into sleep without violating the QoS requirement of the flow.¹ Even when $T_{on \rightarrow off} + T_{off \rightarrow on} < T_{intvl}$, not too much power can be saved unless $T_{on \rightarrow off} + T_{off \rightarrow on} \ll T_{intvl}$. From Fig. 2, we can see that $\frac{T_{on \rightarrow off} + T_{off \rightarrow on}}{T_{intvl}}$ 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 $T_{on \rightarrow off} + T_{off \rightarrow on} \ll T_{intvl}$.

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

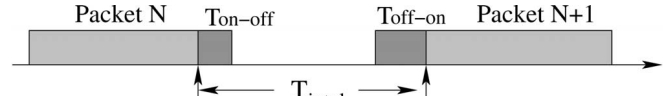


Fig. 2. The relationship between state transition and the packet interval time.

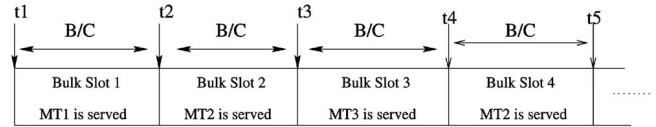


Fig. 3. An example of bulk scheduling.

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. Fig. 3 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 $T_{on \rightarrow off} + T_{off \rightarrow on}$, 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.

3.2 The Rate-Based Bulk Scheduling (RBS) Service Model

In order to address the drawbacks of NVC and BKS, we propose a rate-based bulk scheduling (RBS) service model considering both QoS provision and power efficiency. The basic idea of RBS 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 requirements. To provide QoS to each flow and simplify the implementation, the *start-time first fair queuing (SFQ)* [7] is used to give each flow a fair link sharing. With SFQ, each packet has a start tag and a finish tag. When the j th packet of f_i arrives, its start tag S_i^j and finish tag F_i^j follows:

$$S_i^j = \max\{V(t), F_i^{j-1}\}$$

$$F_i^j = S_i^j + \frac{l_i^j}{r_i},$$

where $V(t)$ is the virtual time of the system and equal to the start time of the packet in service,² l_i^j (in bits) is the packet

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.

2. If the server is idle at t , $V(t)$ is set to the maximum finish tag of the packet that has been served by t .

length and r_i (in bps) is the data rate of f_i . The packets are served in increasing order of the associated start tag. While serving each flow, the scheduler records the service amount of each flow. When a flow has enough *ahead-service*³ with regard to its QoS requirements, the scheduler suspends serving the flow so that the related WNI can sleep and wake up only when the ahead-service is not enough to satisfy the QoS requirements. In other words, the scheduler serves flows that do not have enough ahead-service with the SFQ service discipline and conditionally suspends flows that have enough ahead-service until the buffered data runs out. In this way, the WNI can sleep for many time slots, which are long enough to offset the state transition overhead. As the WNIs of some flows enter sleep, there are less number of active flows in the system (ideally only one flow). Thus, the WNIs only stay in active for a very small amount of time to get enough ahead-service.

The RBS 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 flows. The proxy is used to coordinate with the scheduler and manage the operating state of the WNI based on the amount of buffered data. Next, we describe the details of the scheduler and illustrate how the proxy coordinates with the scheduler.

3.2.1 Balancing Power Efficiency and Fairness

For wireless channels with multirate capability, the amount of bits transmitted per time slot depends on the respective modulation and coding scheme [27]. As a result, there exists a tradeoff between system throughput and fairness among flows. Always serving flows with high transmission rate can improve the system throughput, but may starve flows with low transmission rate. Several schemes have been proposed to opportunistically exploit the system throughput with the constraint of long-term temporal or throughput fairness [10], [14], [13]. However, since these schemes only guarantee long-term fairness among flows, a flow may be starved for a long time (say, a few seconds) [13]. Due to the stringent delay requirement of the streaming applications, the suffering flow may not have enough buffered data to combat the impact of channel errors and cannot maintain the QoS. As a result, long-term fairness is not enough for QoS provision and error resiliency. One simple solution that achieves QoS provision for streaming applications is to adopt traditional rate-based guaranteed service models [33] and provide short-term throughput fairness to each flow. However, because flows with poor channel condition would consume a large portion of the bandwidth, providing short-term throughput may significantly degrade the system throughput [10], [14], [13], [15], [17], [22]. This not only reduces the actual data rate of each flow, but also causes the WNI of each MT to stay longer in active, consuming much more power to receive a certain amount of data.

To achieve power efficiency and error resiliency for each flow, we balance the trade-off between system throughput and short-term fairness by modifying SFQ to provide *temporal fairness* to each flow. In each time slot, the system exploits the highest possible transmission rate for the serving flow according to the channel condition.

Flows with good channel condition can get high power efficiency since they take less time to receive a certain amount of data, while flows with poor channel condition can avoid being starved for a long time. Similar to SFQ, each packet has two tags: the start tag and the finish tag. Unlike SFQ, we use different rules to calculate the finish tag of each packet. Suppose the j th packet of f_i , denoted by p_i^j , is transmitted at the rate of $X_i \in \{C^1, C^2, \dots, C^M\}$, where C^M is the highest available transmission rate. The start tag S_i^j and finish tag F_i^j of p_i^j are defined by:

$$\begin{aligned} S_i^j &= \max\{V(t), F_i^{j-1}\} \\ F_i^j &= S_i^j + \frac{C^M l_i^j}{X_i r_i}. \end{aligned} \quad (1)$$

Compared to SFQ, the normalized service of p_i^j is scaled by $\frac{C^M}{X_i}$, which means that the amount of data transmitted per time slot is accounted as if each flow is served in an error-free channel. In this way, during a given time period, the time slots can be fairly allocated in proportion to the data rate of each flow, whereas more data would be transmitted by the flows with good channel conditions than that with bad channel conditions.

3.2.2 Service Accounting

With RBS, an active flow with enough ahead-service may be suspended by the scheduler for a time period. In order to calculate the ahead-service, each flow f_i is associated with a service counter W_i (in seconds). Suppose f_i is served during the time period of $[t_1, t_2]$. The service counted for f_i during $[t_1, t_2]$, denoted by $W_i(t_1, t_2)$, can be calculated by:

$$W_i(t_1, t_2) = \sum_{j=k}^l \frac{l_i^j}{r_i}, \quad (2)$$

where p_i^k and p_i^l is the first and last packet of f_i served or being served during $[t_1, t_2]$. In practice, t_1 can be the time when the flow starts the session. Equation (2) alone cannot precisely keep track of the buffered service (in seconds) at the client side. Since the buffer size is no less than zero, we need to take into account the time elapsed when the client's buffer is empty. As a result, the obtained ahead-service (in seconds) of f_i during $[t_1, t_2]$, denoted by $ahead_i(t_1, t_2)$, is calculated by:

$$\begin{aligned} ahead_i(t_1, t_2) &= W_i(t_1, t_2) - (t_2 - t_1) \\ &\quad + \int_{t_1}^{t_2} \mathbf{I}(W_i(t_1, s) + t_1 < s) ds, \end{aligned} \quad (3)$$

where $\mathbf{I}(x)$ is 1 if x is true; otherwise it is 0. The integration part is used to compute the time elapsed when the buffer is empty during $[t_1, t_2]$.

3.2.3 Flow State Management

With the accounted service of each flow, the RBS scheduler decides when to serve which flow. At the BS side, each flow has two scheduling states: *idle* and *active*. The transitions between the scheduling states are controlled by the scheduler. For the purpose of flow control (note that each MT usually has limited buffer), there is an upper limit of ahead-service for f_i , denoted by $Maxserv_i$. If $ahead_i > Maxserv_i$,

3. The flow has enough buffered data to provide QoS at the client side.

Notations: \mathcal{A} : the set of flows in active t : the current time S_i, F_i : the start tag and finish tag of the head-of-line packet of f_i

```

1 begin;
2 if ( $\mathcal{A} == NULL$ )
3   { idle in the time slot; goto begin; }
4  $f_i = \arg \min_{f_j \in \mathcal{A}} \{S_j\}$ ;
5  $p = f_i$ .deque(); /* get the packet to be transmitted */
6 if ( $ahead_i > Maxserv_i \vee (ahead_i \geq \phi \wedge \exists j (f_j \in \mathcal{A} \wedge ahead_j < \phi))$ )
7   mark( $p$ );
8 send ( $p$ );
9 if (transmission is successful)
10  { if ( $p$  is marked)
11    {  $state_i = idle$ ;  $e_i = t + ahead_i$ ; }
12  else
13    update  $S_i$  and  $F_i$  according to Eq (1); }
14 goto begin;

```

Fig. 4. The RBS algorithm at the BS.

the scheduler stops serving f_i in order to avoid overflow of the MT's buffer and changes the state of f_i , denoted by $state_i$, to idle. When the scheduler has provided 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) and there exists another active flow, say f_j , which has not got enough ahead-service (i.e., $ahead_j < \phi$), the scheduler suspends serving f_i by changing $state_i$ to idle.

When $state_i$ is set to be idle, the scheduler sets the eligible time of f_i , denoted by e_i , to the current time plus $ahead_i$, which indicates when the scheduler will resume serving f_i . Meanwhile, it notifies MT_i to 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 to represent whether the packet is marked or not. The formal description of the RBS algorithm at the BS is shown in Fig. 4.

3.2.4 Dealing with Channel Errors

Sometimes, the channel condition may be too bad to be tolerated by using modulation and coding schemes. At this time, the probability of a successful transmission becomes low, and bandwidth and power may be wasted during retransmissions. To deal with channel errors, time slots are swapped from flows suffering channel error to flows with good channel conditions. In particular, when f_i has channel errors:

- **If** $ahead_i \geq \phi$, the scheduler suspends f_i and sends a marked null packet to MT_i . When MT_i receives the packet, the proxy shuts down the WNI.
- **Else if** $ahead_i < \phi$, it may not be worth powering off the WNI since the actual sleep time could be too short. Since the time period of some channel errors may be short (especially in the case of fast fading), it is better to postpone serving f_i a little bit. Thus, the RBS scheduler stops serving f_i for a prespecified period, called the *backoff period*, during which the WNI of f_i stays in active. After the backoff period,

the scheduler tries to resume serving f_i . If f_i still suffers from channel errors, the same backoff procedure is applied to f_i again.

For the idle flows in the system, simply turning on the WNI when the buffered data runs out may not be enough to tolerate channel errors. For example, as shown in Fig. 5, suppose the WNI of MT_i is active from t_1 to t_2 and is turned off at t_2 after it has buffered enough data. At t_3 , when the buffer is near empty, the WNI wakes up to receive more data to fill the buffer. At the same time, since the channel condition is continuously bad, most data packets addressed to MT_i will be corrupted and the QoS requirements cannot be met. To alleviate the impact of channel errors on QoS, it should be better to let the WNI wake up earlier than t_3 so that MT_i could have got data under good channel conditions. It is easy to see that this approach may increase the power consumption since the sleep time is reduced. To balance the trade-off between error-resilience and power conservation, we design a novel adaptive scheme to adjust the sleep time of the WNI according to the channel condition.

At any time t , a probabilistic function is used to evaluate the QoS provision of a flow f_i , which is defined as follows:

$$Pr(W_i(t_1, t) - (t - t_1) < \delta_i) \leq \mathcal{P}_i, \quad (4)$$

where δ_i is the threshold of service deficiency of f_i and it is less than zero (e.g. $-20ms^4$), and \mathcal{P}_i is the target probability. For a fixed \mathcal{P}_i , a smaller $|\delta_i|$ provides more stringent delay requirement. For each flow f_i , it has a control parameter θ_i (in second) indicating how early the WNI should be turned on before the buffered data is depleted. The value of θ_i is adjusted as follows: At time t , the observation function of f_i , denoted by \mathcal{O}_i , is defined as:

$$\mathcal{O}_i = \frac{\int_{t_1}^t I(W_i(t_1, s) + t_1 < s + \delta_i) ds}{t - t_1}. \quad (5)$$

In time slot k , θ_i is updated as follows:

$$\theta_i(k+1) = \begin{cases} \min\{\theta_i(k) + (\mathcal{P}_i - \mathcal{O}_i)\delta_i, \theta_{max}\}, & \mathcal{O}_i > \mathcal{P}_i \\ \theta_i(k), & \mathcal{O}_i = \mathcal{P}_i \\ \max\{\theta_i(k) + (\mathcal{P}_i - \mathcal{O}_i)\delta_i, 0\}, & \mathcal{O}_i < \mathcal{P}_i, \end{cases} \quad (6)$$

where θ_{max} is a prespecified system parameter and less than ϕ . With (6), according to the difference between the observed QoS (\mathcal{O}_i) and the target QoS (\mathcal{P}_i), the scheduler adaptively decides how early the WNI should wake up before the buffered data runs out and achieves a good balance between power efficiency and error resiliency. From (6), we can see that θ_i is bounded by $[0, \theta_{max}]$. When f_i is suspended with $ahead_i$ at time t , the eligible time e_i is adjusted to $e_i = t + ahead_i - \theta_i$.

3.2.5 The RBS Proxy

A proxy is associated with the receiver of each flow. The proxy downloads data from the BS, monitors the amount of service, and manages the operation modes of the WNI. The

4. The value of δ_i can be set based on the maximum delay requirement of the application.

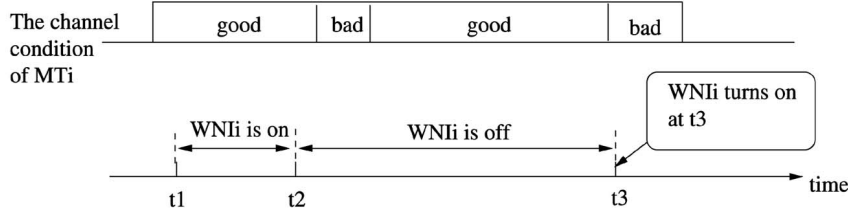


Fig. 5. An example of the impact of channel errors.

proxy coordinates with the server and decides when to activate the WNI. It can calculate the amount of buffered service (in seconds), denoted by $buffer_i$, based on the flow's data rate and the packet length of each buffered packet. If the adaptive scheme for error-resilience is applied, both the BS and the proxy need to run the same algorithm. Otherwise, θ_i is always zero. When the proxy of f_i finds that the received packet is marked, the WNI will sleep for a time period of $\max\{0, buffer_i - \theta_i - transition\ delay\}$. Then, the BS and the proxy agree on the time when the corresponding WNI wakes up so that the BS resumes serving the flow exactly at the time the WNI is in active again. If multiple flows 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* powers it on at any time. Since the MT knows all proxies running on it, these two requirements can be easily implemented. The formal description of the protocol used by the proxy is shown in Fig. 6.

3.2.6 Computation Complexity of RBS

The cost of selecting the serving flow is $O(\log(n))$ and the updates for ahead-service is at the cost of $O(n)$. Thus, RBS is computationally feasible in many wireless networks that have a moderate number of flows per BS. Because most MTs are on the single user basis, in most cases, each user does not have many flows simultaneously and, then, the number of flows is on the same order of the number of users. Thus, the cost of maintaining per-flow state in RBS is not much higher than that of the current per-user-based management at the radio network controller in 3G. Since

Notations:

t_w : the associated wakeup time of the WNI
 t : the current time
 $buffer_i$: the buffered service of f_i (in seconds)

t_w expires

1 turn on the WNI;

On receiving packet p

2 $buffer_i = buffer_i + p.length/r_i$;
 3 if (p is marked \wedge no other proxy needs the WNI)
 4 $\{ t_w = t + \max(0, buffer_i - \theta_i - transition\ delay);$
 5 power off the WNI; }

On consuming packet p

6 $buffer_i = buffer_i - p.length/r_i$;

most scheduling modules are implemented by software or FPGA [9], new scheduling algorithms can be accommodated. MTs can easily setup the data rate of the flow, ϕ , Maxserv, the backoff period during session initialization. The transition delay of the WNI can be obtained from the manufacturer and ϕ is simply required to be much larger than the transition delay.

3.3 Analysis of RBS

In this section, we prove that RBS can provide the delay guarantee and achieve power efficiency for all flows in the system. For simplicity, we assume the channel is error-free and the *Maxserv* of each flow is infinite. Since the channel is error-free, the modified SFQ is the same as the original one. We first present the delay guarantee of RBS. The *expected arrival time* of packet p_i^j served by an RBS server is defined as:

$$EAT(p_i^j) = EAT(p_i^k) + \sum_{m=k}^{j-1} \frac{l_i^m}{r_i}, j > k \geq 0, \quad (7)$$

where p_i^k is the last packet served before flow f_i is activated.

Theorem 1. *If Q is the set of all backlogged flows in service by a RBS server and $\sum_{n \in Q} r_n \leq C$, the RBS scheduler will serve each packet p_i^j conforming to:*

$$d_i^j - EAT(p_i^j) \leq \frac{|Q|L^{max}}{C}, \quad (8)$$

where d_i^j is the departure time of p_i^j and L^{max} is the maximum packet length.

Proof. The set Q can be partitioned into two subsets: the set of the 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 $EAT(p_i^j) > t$ (otherwise $f_i \in \mathcal{A}$) and $d_i^j \leq t$, the theorem holds.
- **Case 2:** $f_i \in \mathcal{A}$. Suppose the last time f_i was activated is t_a . Since f_i is served with SFQ during the interval $[t_a, t]$ with the same arguments of Theorem 4 in [7] and $\mathcal{A} \subseteq Q$, we get:

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

□

Fig. 6. The protocol used by MT_i 's proxy.

Lemma 1. Suppose Q is the set of backlogged flows being served by a SFQ server during $[t_1, t_2]$, $\forall(n \in Q)ahead_n(t_1) = 0$ and $\sum_{n \in Q} r_n < C$, the following property holds for f_i :

$$\begin{aligned} & \left(\frac{C - \sum_{n \in Q} r_n}{\sum_{n \in Q} r_n} \right) (t_2 - t_1) - \frac{|Q|L^{max}}{\sum_{n \in Q} r_n} \leq ahead_i(t_1, t_2) < \\ & \left(\frac{C - \sum_{n \in Q} r_n}{\sum_{n \in Q} r_n} \right) (t_2 - t_1) + \left(\frac{1}{r_i} + \frac{|Q| - 1}{C} \right) L^{max}. \end{aligned} \quad (9)$$

Proof. By adopting the concept of the rate controller in [2], the actual data rate of f_i , denoted by \hat{r}_i , is equal to

$$\hat{r}_i = \frac{r_i}{\sum_{n \in Q} r_n} C.$$

With the concept of the latency-rate server [25] and Theorem 1, under SFQ, the aggregated service (in bits) of f_i during $[t_1, t_2]$, denoted by $S_i(t_1, t_2)$, follows:

$$S_i(t_1, t_2) \geq \hat{r}_i(t_2 - t_1) - \frac{\hat{r}_i |Q| L^{max}}{C}. \quad (10)$$

Meanwhile, since the SFQ scheduler serves the packet with the minimum start tag, which must be eligible to be served in the corresponding GPS system, with Theorem 1 of [2], we get:

$$S_i(t_1, t_2) \leq \hat{r}_i(t_2 - t_1) + \left(1 - \frac{\hat{r}_i}{C} \right) L^{max}. \quad (11)$$

Since $S_i(t_1, t_2) = W_i(t_1, t_2) * r_i$ and $\hat{r}_i = \frac{C}{\sum_{n \in Q} r_n} r_i$, combining (10) and (11), we have:

$$\begin{aligned} & \frac{C}{\sum_{n \in Q} r_n} (t_2 - t_1) - \frac{|Q|L^{max}}{\sum_{n \in Q} r_n} \leq W_i(t_1, t_2) < \\ & \frac{C}{\sum_{n \in Q} r_n} (t_2 - t_1) + \left(\frac{1}{r_i} - \frac{1}{C} \right) L^{max}. \end{aligned} \quad (12)$$

With (10), we get:

$$0 \leq \int_{t_1}^{t_2} \mathbf{I}(W_i(t_1, s) - t_1 < s) ds \leq \frac{|Q|L^{max}}{C}. \quad (13)$$

With (3), (12), and (13), by simple conversions, the Lemma holds. \square

Lemma 2. Suppose Q is the set of backlogged flows being served by an RBS server from t_1 , $\forall(i \in Q)ahead_i(t_1) = 0$, and $\sum_{n \in Q} r_n < C$. If ϕ satisfies

$$\forall(i \in Q), \phi > \frac{\sum_{n \in Q} r_n \left(\frac{1}{r_i} + \frac{|Q| - 1}{C} \right) L^{max}}{C - \sum_{n \in Q} r_n},$$

there are at most $|Q| - 1$ flows sharing the channel together after

$$t_1 + \frac{\sum_{n \in Q} r_n \phi}{C - \sum_{n \in Q} r_n}.$$

Proof. Since $Maxserv$ is infinite for each flow, according to the RBS scheme, the RBS server is always busy. Thus,

during any time interval $[t_1, t]$, $C(t - t_1) = \sum_{i \in Q} S_i(t_1, t)$ holds. Since

$$\begin{aligned} \forall(i \in Q), S_i(t_1, t) & \leq W_i(t_1, t) r_i + \int_{t_1}^t \mathbf{I}(W_i(t_1, s) - t_1 < s) ds \\ & = (ahead_i(t_1, t) + (t - t_1)) r_i, \end{aligned}$$

we get:

$$\sum_{n \in Q} ahead_n(t_1, t) r_n \geq \left(C - \sum_{n \in Q} r_n \right) (t - t_1). \quad (14)$$

Suppose at $t^* (t^* > t_1)$, f_i is the first flow that gets the ahead-service of at least ϕ , with Lemma 1 and (14), we get

$$t^* > t_1 + \frac{\phi - \left(\frac{1}{r_i} + \frac{|Q| - 1}{C} \right) L^{max}}{\frac{C - \sum_{n \in Q} r_n}{\sum_{n \in Q} r_n}}. \quad (15)$$

Now, we prove the Lemma by contradiction. Suppose at time $\tilde{t} (\tilde{t} > t^*)$, all flows are served together again. With RBS, the ahead-service of each flow must be less than ϕ , we have:

$$\tilde{t} < t_1 + \frac{\phi}{\frac{C}{\sum_{n \in Q} r_n} - 1}. \quad (16)$$

Since

$$\phi > \frac{\sum_{n \in Q} r_n \left(\frac{1}{r_i} + \frac{|Q| - 1}{C} \right) L^{max}}{C - \sum_{n \in Q} r_n}$$

and $e_i \geq t^* + \phi$, with (15) and (16), we have $\tilde{t} - e_i < 0$, which contradicts the assumption that all flows are served at \tilde{t} . \square

Lemma 3. Suppose a backlogged flow f_i is served by a RBS server. During each period when f_i is in active state, the total amount of transmitted data (in bits) is no less than $\frac{\phi r_i C}{C - r_i}$.

Proof. Suppose flow f_i receives B_k^i bits during the active period of Δt_i . With RBS, f_i is suspended only after it has the ahead-service of at least ϕ . Then, we have

$$B_k^i \geq \left(\Delta t_i + \phi - \int_{t_{a,i}^k}^{t_{a,i}^k + \Delta t_i} \mathbf{I}(W_i(t_{a,i}^k, s) - t_{a,i}^k < s) ds \right) r_i, \quad (17)$$

where $t_{a,i}^k$ is the time when f_i starts to transmit B_k^i . As explained in Lemma 1, $\int_{t_{a,i}^k}^{t_{a,i}^k + \Delta t_i} \mathbf{I}(W_i(t_{a,i}^k, s) - t_{a,i}^k < s) ds$ is bounded by $\frac{|Q|L^{max}}{C}$, we can see that B_k decreases as Δt_1 decreases. Since Δt_i reaches the minimum value when f_i is served alone during Δt_i , we have $\Delta t_i^{min} = \frac{\phi r_i}{C - r_i}$. Since Δt_i^{min} is achieved when f_i is served alone, we have $B_k^i \geq \left(\frac{\phi r_i}{C - r_i} + \phi \right) r_i = \frac{\phi r_i C}{C - r_i}$. \square

Theorem 2. Suppose Q is the set of backlogged flows being served by an RBS server from t_1 , $\forall(i \in Q)ahead_i(t_1) = 0$ and $\sum_{n \in Q} r_n < C$. Assume the following two conditions hold to transmit B bits ($B \gg \phi C$):



Fig. 7. An illustration of data transmission under RBS.

1.

$$\forall (i \in Q), \phi > \frac{\sum_{n \in Q} r_n (\frac{1}{r_i} + \frac{|Q|-1}{C}) L^{max}}{C - \sum_{n \in Q} r_n}, \quad (18)$$

2.

$$\begin{aligned} &\forall (i \in Q), \\ &\left(\frac{\phi r_{min}}{C - r_i} - \frac{(|Q| - 1) L^{max}}{C} - \frac{\sum_{n \in Q} r_n (\frac{1}{r_i} - \frac{1}{C}) L^{max}}{C} \right) \\ &P_a > E_s, \end{aligned} \quad (19)$$

where P_a (in Watt) is the power consumption in active, E_s (in Joule) is the energy used by a state transition, and $r_{min} = \min_{n \in Q} \{r_n\}$. Then, RBS is more power efficient than SFQ.

Proof. As shown in Fig. 7, for a flow, say f_i , served by an RBS server, the total data of B is transmitted in the form of several runs of data (i.e., $\sum_{k=1}^N B_k = B, N > 2$). With condition 1 and Lemma 2, f_i can be served with the actual data rate of at least $\frac{r_i}{\sum_{n \in Q} r_n - r_{min}} C$ during the transmission of $B_k (k = 2 \dots N)$. With Lemma 3, we have $B_k \geq \frac{\phi r_i C}{C - r_i}$.

Now, let's compare the power consumption used by the WNI for receiving B data between RBS and SFQ. With (10), we have the time spent by f_i to transmit B data in SFQ, denoted by $T_i^{SFQ}(B)$, follows:

$$T_i^{SFQ} \geq \frac{B - (1 - \frac{r_i}{C}) L^{max}}{\sum_{n \in Q} r_n C}. \quad (20)$$

Thus, the energy (in Joule) used by the WNI of f_i for receiving B data in SFQ, denoted by $E_i^{SFQ}(B)$, follows:

$$E_i^{SFQ}(B) \geq \frac{B - (1 - \frac{r_i}{C}) L^{max}}{\sum_{n \in Q} r_n C} P_a.$$

With RBS, according to (10), condition 1 and Lemma 2, the maximum time to transmit B_k data is less than or equal to

$$\frac{(\sum_{n \in Q} r_n - r_{min}) B_k}{r_i C} + \frac{(|Q| - 1) L^{max}}{C} (k = 2 \dots N).$$

The power consumed by the WNI for receiving B_1 data in RBS is less than or equal to that in SFQ. Since

5. It is possible that B_N does not have this property. In this case, without loss of correctness, we discard B_N by setting $N = N - 1$.

$B = \sum_{i=1}^N B_i$, the power consumption difference between SFQ and RBS follows:

$$\begin{aligned} &E_i^{SFQ}(B) - E_i^{RBS}(B) \geq \\ &\sum_{k=2}^N \left(\frac{\sum_{n \in Q} r_n B_k}{r_i C} P_a - E_i^{RBS}(B_k) - E_s \right) \\ &- \frac{\sum_{n \in Q} r_n (\frac{1}{r_i} - \frac{1}{C}) L^{max}}{C} P_a \geq \sum_{k=2}^N \\ &\left(\left(\frac{\sum_{n \in Q} r_n \phi}{C - r_i} - \frac{(\sum_{n \in Q} r_n - r_{min}) \phi}{C - r_i} - \frac{(|Q| - 1) L^{max}}{C} \right) P_a - E_s \right) \\ &- \frac{(\sum_{n \in Q} r_n) (\frac{1}{r_i} - \frac{1}{C}) L^{max}}{C} P_a \geq \sum_{k=2}^N \\ &\left(\left(\frac{\phi r_{min}}{C - r_i} - \frac{(|Q| - 1) L^{max}}{C} \right) P_a - E_s \right) \\ &- \frac{(\sum_{n \in Q} r_n) (\frac{1}{r_i} - \frac{1}{C}) L^{max}}{C} P_a. \end{aligned}$$

With condition 2, we have $E_i^{SFQ}(B) > E_i^{RBS}(B)$. \square

Remark. Theorem 2 gives the sufficient conditions to guarantee that RBS is more power efficient than SFQ. In terms of power consumption, referring to the calculation of $T_i^{SFQ}(B)$, we can see that the power consumption of other rate-based service models [30], [33] is similar to SFQ. Thus, the RBS model is also more power efficient than other rate-based service models.

4 PERFORMANCE EVALUATIONS

4.1 The Experimental Setup and Parameters

We evaluate the performance of RBS through a case study called *Audio-on-Demand (AoD)*. The AoD streaming service is evaluated through trace-driven simulations. We downloaded a demo MP3 audio stream from [19] as the trace source. Since the MP3 streams are based on variable bit rate (VBR), we extracted the information of every frame, i.e. frame size, frame bit rate, frame sample rate, etc., to get the bit rate of each packet. We assume that the start time of each flow is uniformly distributed in $[0.0, 2.0]$ and the simulation time is 100 seconds.

We assume an EDGE [27] system in our simulations, where the AoD service is carried in a dedicated channel, called AoD channel. The channel is accessed based on TDMA. In order to evaluate the impact of different transmission rates, based on the results of [28], a five-state Markov chain is used to emulate the process of the channel condition with fast fading when the channel is error-prone. As shown in Fig. 8, the marked line or curve shows the transition probability from one state to another. The transmission rate in state 4 is equal to the capacity of the AoD channel, which is assumed to be $384Kbps$, and is zero when the channel condition is in state 0. Following the standard of EDGE [27], each time slot is $2.5ms$, which can be used to transmit 112, 74, 56, 44 bytes of data (not including

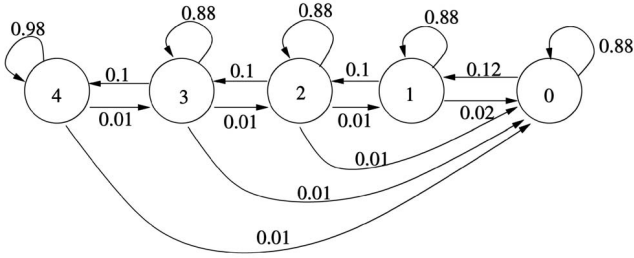


Fig. 8. The error-prone channel model.

the header) in states 4, 3, 2, 1, respectively.⁶ Each AoD flow is assigned a data rate of $56Kbps$. With the result of [23], we assume $T_{on \rightarrow off} = 5ms$ and $T_{off \rightarrow on} = 10ms$ for each WNI. The RBS server sets $\phi = 80ms$ and $MaxServ = 3,000ms$.

We monitor the behavior of RBS by the trace of the buffer size at each MT and evaluate the performance of RBS based on the following factors: the time spent in each operation mode and the quality of the playback audio. Because the power consumption in active is much larger than that in sleep and the power consumption of state transition is on the same order of that in active [29], the power efficiency of each scheme can be reflected by the sum of the time spent in active and the time spent in transition. We use the *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. We compare the performance of RBS with the bulk scheduling (BKS) and the Nonwork-conserving Virtual Clock (NVC) algorithms. We choose the length of the bulk slot to be 1.0 second for BKS. There is a buffer on the MT side in BKS and NVC. For BKS, the MT stays in sleep after being served until the buffered data runs out. Because each bulk slot is allocated to a flow in a nonpreemptive way, the BKS scheduler cannot take into account of the channel condition. Different from BKS, NVC performs channel state-aware scheduling. Similar to RBS, flows with channel errors in NVC also yield the channel, and back off for a backoff period, which is set to be $15ms$ for both RBS and NVC. With NVC, the WNI is powered off when the packet interval is greater than $T_{on \rightarrow off} + T_{off \rightarrow on} + 5ms$ or when the buffered service is greater than $Maxserv$. The evaluation considers three scenarios. In Scenario 1, the channel is error-free. In Scenario 2, an error-prone channel is considered. We evaluate the impacts of ϕ on the performance of RBS in Section 4.4 and study the impacts of transition delay in Section 4.5.

4.2 Scenario 1: Error-Free Channel

In this scenario, five identical AoD flows in an error-free channel are used to show the fundamental differences among RBS, BKS, and NVC. From Fig. 9a, we can see that the playback quality of RBS and NVC are perfect since their total NITs are 0. However, BKS violates the delay requirement many times. For example, the total NITs of all flows are greater than 3 seconds. This can be explained as follows: Since there are five flows sharing the channel, it is highly possible that more than one flows request data at the

beginning of a bulk slot. Since BKS only randomly selects one winning flow to serve, other losing flows cannot be served during the bulk slot. Because the bulk slot is 1.0 second, all the losing flows are starved for 1.0 second.

Although the playback quality under NVC is perfect, from Fig. 9b, we can see that the power consumption of the WNI under NVC is significantly larger than that under RBS. This is due to the fact that the data rate of the flow is quite high and many of the interpacket arrival periods are less than $T_{on \rightarrow off} + T_{off \rightarrow on}$. As a result, the WNI cannot frequently go to sleep in the NVC approach. Even though the WNI can enter sleep, the actual sleep period is significantly reduced by $T_{on \rightarrow off} + T_{off \rightarrow on}$. For example, if the data rate is $56Kbps$ and the maximum payload of a packet is 112 Bytes, the interpacket arrival time is $16ms$. Then, the actual sleep period is only $1ms$ and it is not worth to power off the WNI. Note that no data will be received during state transition.

Fig. 9c shows how these three approaches work through the buffer trace. When the buffer size goes up, the WNI receives data in the active mode. The slope of 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. During many time periods, the slope in NVC is much less than that in RBS, which means that the flow takes more time to get a certain amount of data in NVC than in RBS. This is because the RBS scheduler suspends flows with enough ahead-service so that the remaining active flows can have much higher actual data rate. In NVC, since the interpacket arrival period is too short, the flow has to keep active and compete with other flows for the channel. As a result, the actual data rate of the flow is not as high as that in RBS.

4.3 Scenario 2: Error-Prone Channel

In this section, we evaluate the impacts of channel errors. To evaluate the effect of temporal fairness, we compare RBS and the *RBS-O* scheme, which applies the original SFQ [7] and provides short-term throughput fairness to each active flow. For RBS, we also study the performance of the error-resilient enhancement (see the adaptive scheme in Section 3.2.4) and denote it as *RBS-E*. We introduce channel errors in this scenario. Specifically, we assume flows 1-3 have channel error and the channel of flows 4 and 5 is error-free. For each flow, \mathcal{P}_i , δ_i , and θ_{max} are set to be 0.01 , $-10ms$, and $40ms$, respectively.

As shown in Fig. 10a, the playback quality of each flow in BKS is much worse than that in RBS. The poor playback quality of BKS has been explained in Scenario 1 and the reason is still valid for this example. Comparing RBS and *RBS-O* in Figs. 10a and 10b, we can see that, for each flow, RBS saves much more power than *RBS-O* without loss of playback quality. This verifies that RBS can achieve a good balance between power efficiency and fairness.

With the backoff mechanism for flows with channel error, the playback quality of flows 4 and 5 in RBS and NVC are not affected by channel errors. However, NVC results in much higher power consumption than RBS. Moreover, the playback qualities of flows 1-3 in NVC are much worse than those in RBS. For NVC, as explained in Scenario 1, there are always a large number of flows sharing the channel and each flow has low data rate. Meanwhile, the system throughput is degraded due to channel errors. As a result, the actual

6. Because EDGE and HDR/EV-DO both use rate adaptive techniques, we can also evaluate the performance of RBS in HDR/EV-DO by changing the Markov chain and the transmission rate in each state.

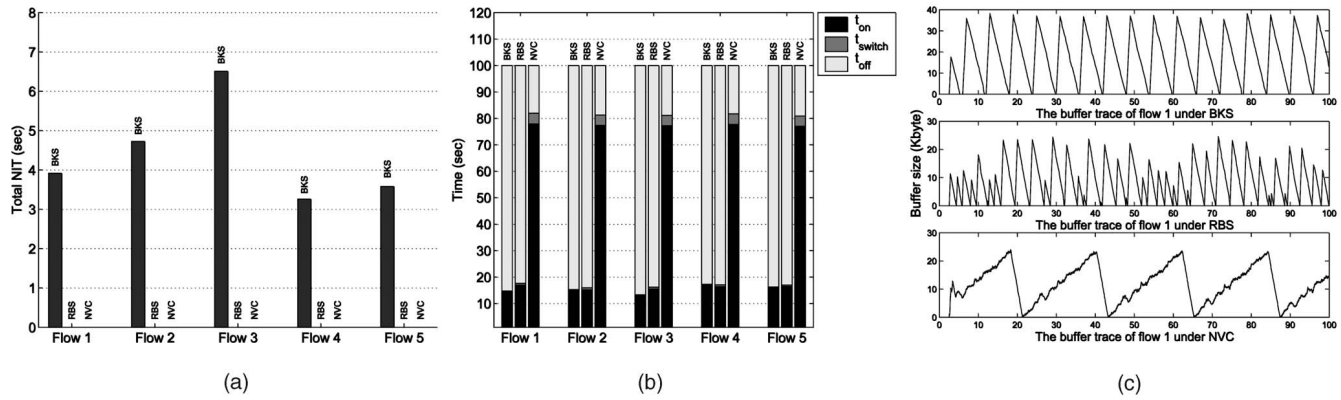


Fig. 9. Performance comparisons among BKS, RBS, and NVC under in error-free channel. (a) Playback quality. (b) Power consumption. (c) Buffer trace.

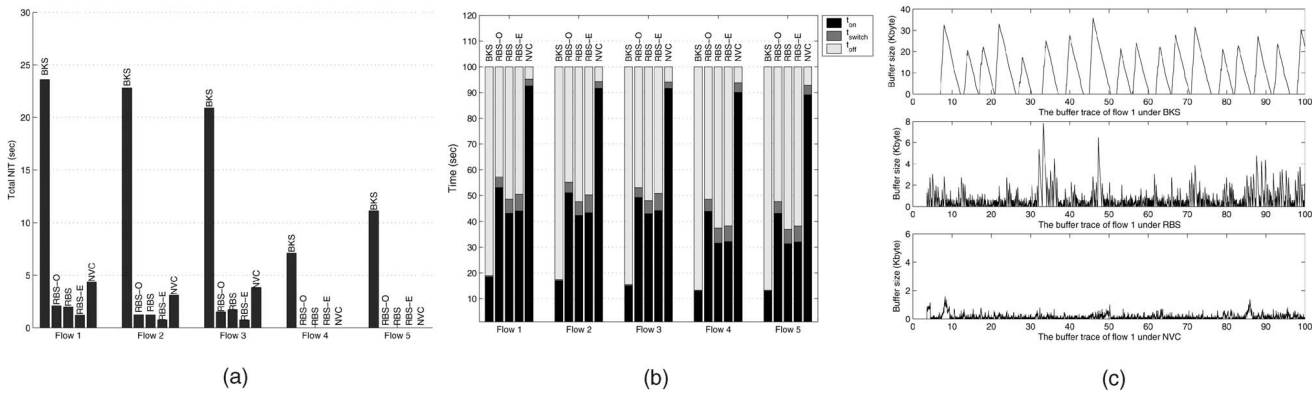


Fig. 10. Performance comparisons among BKS, RBS, RBS-O, RBS-E, and NVC in error-prone channel. (a) Playback quality. (b) Power consumption. (c) Buffer trace.

data rate of the flow could be too low to provide QoS sometimes. In RBS, at a time, the number of flows sharing the channel could be much smaller than in NVC.

As a result, on the average, each flow can buffer data faster than in NVC. With more buffered data, the playback quality of each flow in RBS is better than that in NVC.

From Fig. 10b, we can see that the power consumptions in BKS, RBS, and NVC become larger than those in Scenario 1. This is mainly caused by the decreased system throughput due to channel errors. Comparing BKS and RBS, we can see that the difference of the power consumption is greater than that in the error-free channel. For example, t_{switch} of each flow in RBS increases. Since the actual data rate of each flow decreases, on average, the amount of buffered data is reduced. This can be easily verified by comparing the amount of buffered data in Fig. 9c and Fig. 10c.

To deal with channel error, we use the error-resilient enhancement to control how long the WNI should sleep based on the channel condition. From Fig. 10a, we can see that RBS-E provides better playback quality than RBS. On average, the total NIT of each flow with channel errors can be reduced by 50 percent. By turning on the WNI earlier, the flow could have more chance to buffer more data to tolerate long period of error. As shown in Fig. 10b, t_{on} in RBS-E is almost the same as that in RBS, but t_{switch} in RBS-E is more than that in RBS. When the WNI wakes up before the buffered data runs out, its sleep time is reduced, which increases the number of state transitions. Since we use the

adaptive scheme to carefully adjust the time to power on the WNI according to the channel condition, the power consumption of the WNI in RBS-E is slightly higher than that in RBS.

4.4 The Impacts of ϕ

We evaluate the performance with three different values of ϕ : 40ms, 80ms, and 180ms. The simulation settings are the same as Scenario 2, except that θ_{max} is set to be 20ms when $\phi = 40ms$. As ϕ increases, the number of state transitions drops. However, if ϕ is too large, the time spent in getting enough ahead-service could be much longer. Fig. 11a shows the power consumption of the WNI of flow 1 with different values of ϕ . As expected, for RBS-O, RBS, and RBS-E, when ϕ increases, the time to get enough ahead-service increases and, then, t_{on} increases. On the other hand, as ϕ decreases, the number of state transitions increases, which leads to increased t_{switch} . In particular, the impact of ϕ on the power consumption in RBS-O is larger than that in RBS and RBS-E, which indicates the drawback of short-term throughput fairness with regard to power efficiency. Overall, ϕ cannot be set too small or too large. If ϕ is too small (large), t_{switch} (t_{on}) becomes quite large.

We also study the impact of ϕ on playback quality. As shown in Fig. 11b, in RBS-O, RBS, and RBS-E, the playback quality of flow 1 increases as ϕ increases. On average, before the flow has enough ahead-service, the amount of buffered data of the flow increases as ϕ increases. When the flow suffers from channel errors before it has enough ahead-

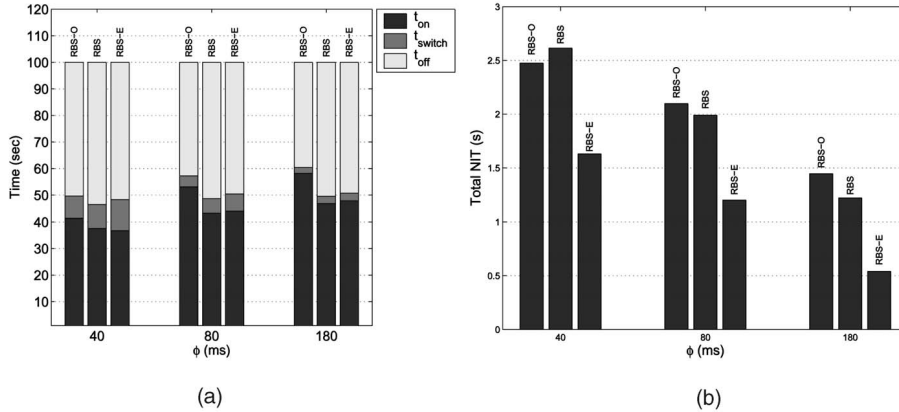


Fig. 11. Performance of RBS with different ϕ . (a) Power consumption. (b) Playback quality.

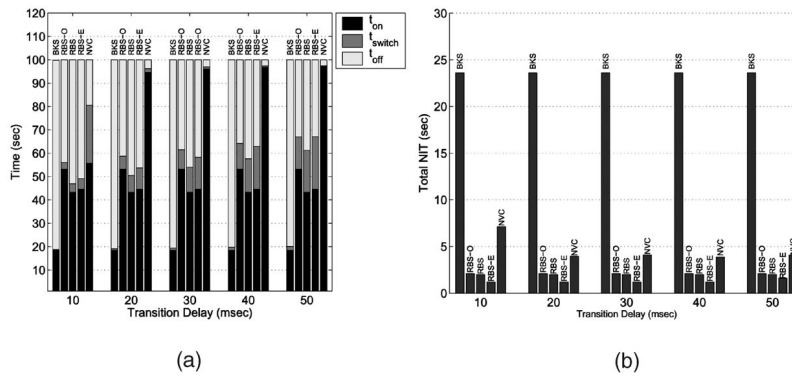


Fig. 12. Performance comparisons among BKS, RBS, RBS-O, RBS-E, and NVC under different transition delays. (a) Power consumption. (b) Playback quality.

service, it may have more buffered data with a larger ϕ , and be able to tolerate longer period of error. As a result, ϕ cannot be set too small. Otherwise, the playback quality may be significantly degraded. Similar to Scenario 2, in RBS-E, the playback quality of the flow is improved at the cost of a slightly higher power consumption.

4.5 The Impacts of Transition Delay

In this section, we evaluate the effects of transition delay on five scheduling algorithms BKS, RBS, RBS-O, RBS-E, and NVC. We use the same setting as that in Scenario 2 (Section 4.3). As shown in Fig. 12a, when the transition delay increases, the power consumption of BKS does not change too much. This is because each MT in BKS only needs to turn on its WNI at the beginning of a bulk slot, and then the total number of state transitions is quite small (Note that the size of a bulk slot is 1.0 second). NVC behaves differently as the transition delay changes. When the transition delay is very small (e.g. 10ms), NVC can reduce the t_{on} since many packet intervals could be long enough for the MT to power off its WNI without violating the delay requirements. However, due to the large number of state transitions, NVC has a large t_{switch} under such a situation and cannot save much power. When the delay is large, as explained in Scenario 2, WNIs cannot be frequently powered off, resulting in high power consumption. RBS, RBS-O, and RBS-E are more power efficient than NVC in all cases, and RBS-O underperforms RBS and RBS-E. The reasons have been described in Scenario 2 and are still valid here. As the transition delay increases, the total power

consumption of RBS, RBS-O, and RBS-E increases. Although the total power consumption increases, t_{on} does not increase since ϕ is fixed. Thus, the power consumption is increased due to the increased transition delay (t_{switch}).

From Fig. 12b, we can see that the transition delay does not affect the playback quality of BKS, which is the worst among these five algorithms. When the transition delay is very small (e.g., 10ms), the playback quality of NVC is poor. This is due to the fact that if the WNI is frequently powered off, the MT may not have enough buffered data to combat channel errors when the WNI is turned on. Similar to Scenario 2, RBS, RBS-O, and RBS-E have much better playback quality than BKS and NVC. From Fig. 12b, we can see that the transition delay does not change the playback quality of RBS and RBS-O. However, when the transition delay is equal to 50ms, the playback quality of RBS-E degrades a little bit, i.e., the total NIT is increased from 1.2 to 1.4 seconds. Suppose the MT is suspended with the ahead-service of 80ms and it needs to wake up with 40ms buffered service. If the transition delay is 50ms, when the WNI is turned on, its actual buffered service becomes $80 - 50 = 30ms$ rather than 40ms. If the MT suffers from long period of channel errors, the playback quality will be degraded. Since this kind of case does not occur frequently, the quality degradation is not significant. According to the adaptive algorithm in Section 3.2.4, it is easy to see that the transition delay does not affect the playback quality of RBS-E when $\phi - \theta_{max}$ is greater than the transition delay.

5 RELATED WORK

Recently, researchers start to investigate power management issues in wireless networks. For example, IEEE 802.11 [8] supports a 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.

Yuan and Nahrstedt [32] proposed a sender-buffering approach for video sensors. With the sender-buffering approach, each sensor buffers the encoded frames and transmits them in bursts. As a result, it can prolong the WNI sleeping time but slow down the CPU speed. Meanwhile, Chandra and Vahdat [3] proposed a proxy-buffering approach to support power-efficient multimedia playback in wireless LANs. The local proxy shapes the traffic from the access point to the client in bursts and informs the client of the next packet arrival time using a special control packet. Compared to the RBS scheme, these two schemes may incur a long delay to a flow because their scheduling algorithms cannot provide guaranteed service to each flow. Similar to the BKS scheme, when multiple MTs share the channel at the same time, they cannot support streaming applications in wireless networks without significantly degrading playback quality.

Zhang et al. [34] proposed a frame-based scheduling scheme, called the scheduled contention free burst (S-CFB), to achieve energy efficient data transmission with delay guarantees. With the predefined schedule, WNIs can be turned off when they are not in use. However, if the message length varies during each transmission burst [34], bandwidth may be wasted since the burst is not preemptive. Moreover, when considering the impact of channel errors, it becomes more difficult to balance the tradeoff between the length of the transmission burst and the bandwidth utilization. Different from S-CFB, RBS serves each flow on the granularity of packet, which is more flexible in terms of variable packet length and dynamic packet arrival pattern. Also, RBS provides mechanisms to deal with channel errors.

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.

Prabhakar et al. [20] studied power conservation with regard to scheduling. They show that the power consumption can be significantly reduced by lowering 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.

Fitzek and Reisslein [6] proposed a prefetching protocol to support high performance streaming applications over wireless links. Part of the ongoing media streams are prefetched into buffers of the 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 RBS is similar to JSQ. However, RBS focuses on the aspect of power conservation of WNIs, whereas JSQ deals with channel errors of wireless links. Furthermore, RBS also considers achieving power efficient data transmission through scheduling.

6 CONCLUSIONS

In this paper, we proposed a rate-based bulk scheduling (RBS) service model to save power and ensure QoS provision. The RBS scheduler decides to serve or suspend a flow based on the amount of the buffered data. The flow with insufficient buffered data will be served with the start-time first queuing service discipline, whereas the flow with enough buffered data will be suspended until the buffered data runs out. 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. A novel adaptive scheme has been proposed to enhance the error-resilience of RBS by adjusting the sleep time of the WNI according to the channel condition of the flow. Through analysis, we proved that the service model has delay guarantee and is more power efficient than other rate-based fair queuing service models. Simulation results showed that RBS can significantly reduce the power consumption of the WNI and provide QoS when compared to NVC and BKS. Although we only discussed AoD in this paper, it is easy to see that the proposed techniques can be extended to other streaming services. In the future, we also plan to extend RBS to Web and FTP applications. Since these applications have a loose delay requirement, the server has more flexibility to decide when to serve which flow, and when to shutdown which WNI. When multiple flows are associated with the same MT, it would be

interesting to further improve the power efficiency of RBS by coordinating the corresponding proxies.

ACKNOWLEDGMENTS

The authors would like to thank the editor and the anonymous referees whose insightful comments helped improve the quality of the paper. This work was supported in part by the US National Science Foundation (CAREER CCR-0092770 and ITR-0219711).

REFERENCES

- [1] P. Bender, P. Black, M. Grob, R. Padovani, N. Nagabushana, and A. Viterbi, "CDMA/HDR: A Bandwidth-Efficient High-Speed Wireless Data Service for Momadic Users," *IEEE Comm. Magazine*, July 2000.
- [2] J. Bennett and H. Zhang, "WF2Q: Worst-Case Fair Weighted Fair Queueing," *Proc. IEEE INFOCOM*, Mar. 1996.
- [3] S. Chandra and A. Vahdat, "Application-Specific Network Management for Energy-Aware Streaming of Popular Multimedia Formats," *Proc. USENIX Ann. Technical Conf.*, 2002.
- [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 Trans. 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_plugin_gedin_1.html, 2002.
- [6] F.H. Fitzek and M. Reisslein, "A Prefetching Protocol for Continuous Media Streaming in Wireless Environments," *IEEE J. Selected Areas in Comm.*, vol. 19, no. 10, Oct. 2001.
- [7] P. Goyal, H. Vin, and H. Cheng, "Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," *Proc. ACM SIGCOMM*, Aug. 1996.
- [8] IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Spec," *IEEE 802.11 Standard*, 1999.
- [9] K. Ito, T. Kumagai, K. Harada, T. Sonobe, T. Tomita, and E. Ikeda, "Radio Network Control System," *Fujitsu Science and Technology J.*, Dec. 2002.
- [10] F. Kelly, "Charging and Rate Control for Elastic Traffic," *The European Trans. Telecomm.*, vol. 2, 1997.
- [11] R. Krashinsky and H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Shutdown," *Proc. ACM Mobicom*, 2002.
- [12] Q. Li, J. Aslam, and D. Rus, "Online Power-Aware Routing in Wireless Ad-Hoc Networks," *Proc. ACM Mobicom*, July 2001.
- [13] X. Liu, E.K.P. Chong, and N.B. Shroff, "Transmission Scheduling for Efficient Wireless Utilization," *Proc. IEEE INFOCOM*, 2001.
- [14] Y. Liu, S. Gruhl, and E. Knightly, "WCFO: An Opportunistic Wireless Scheduler with Statistical Fairness Bounds," *IEEE Trans. Wireless Comm.*, vol. 2, no. 5, Sept. 2003.
- [15] S. Lu, T. Nandagopal, and V. Bharghavan, "A Wireless Fair Service Algorithm for Packet Cellular Networks," *Proc. ACM Mobicom*, Oct. 1998.
- [16] Motorola, "Motorola V 60i User Manual," <http://commerce.motorola.com/consumer/QWhtml/manual.html>, 2005.
- [17] T.S.E. Ng, I. Stoica, and H. Zhang, "Packet Fair Queueing Algorithms for Wireless Networks with Location-Dependent Errors," *Proc. IEEE INFOCOM*, Mar. 1998.
- [18] Nokia, "Nokia 5510 Specifications," <http://www.nokia.com/phones/5510/specifications.html>, 2005.
- [19] MPEG Org, "MP3 Test Streams," <http://www.mpeg.org/MPEG/mp3.html>, 2005.
- [20] B. Prabhakar, E. Uysal-Biyikoglu, and A. El Gamal, "Energy-Efficient Transmission over a Wireless Link via Lazy Packet Scheduling," *Proc. IEEE INFOCOM*, Mar. 2001.
- [21] Qualcomm, "MSM6500 Chipset Solution," http://www.cdma.tech.com/solutions/pdf/msm6500_chipset.pdf, 2005.
- [22] S. Shakkottai and R. Srikant, "Threshold-Based Dynamic Replication in Large-Scale Video-on-Demand Systems," *Wireless Networks*, vol. 8, pp. 13-26, 2002.
- [23] T. Simunic and S. Boyd, "Managing Power Consumption in Networks on Chips," *Proc. ACM DATE*, 2002.
- [24] M. Stemm and R.H. Katz, "Measuring and Reducing Energy Consumption of Network Interfaces in Handheld Devices," *IEICE Trans. Comm.*, vol. E80-B, no. 8, Aug. 1997.
- [25] D. Stiliadis and A. Varma, "Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms," *IEEE/ACM Trans. Networking*, vol. 6, Oct. 1998.
- [26] V. Vanghi, A. Damnjanovic, and B. Vojcic, *The cdma2000 System for Mobile Communications*. Upper Saddle, N.J.: Prentice Hall, 2004.
- [27] B.H. Walke, *Mobile Radio Networks: Networking, Protocols, and Traffic Performance*, second ed. John Wiley & Sons, Ltd., 2002.
- [28] H.S. Wang and N. Moayeri, "Finite-State Markov Channel—A Useful Model for Radio Communication Channels," *IEEE Trans. Vehicular Technology*, vol. 44, no. 1, pp. 163-171, Feb. 1995.
- [29] A. Wang, S. Cho, C.G. Sodini, and P. Chandrakasan, "Energy Efficient Modulation and MAC for Asymmetric RF Microsensor System," *Proc. Int'l Symp. Low Power Electronics and Design*, 2001.
- [30] J. Xu and R.J. Lipton, "On Fundamental Tradeoff between Delay Bounds and Computational Complexity in Packet Scheduling Algorithm," *Proc. ACM SIGCOMM*, Aug. 2002.
- [31] Y. Xu, J. Heidemann, and D. Estrin, "Geography-Informed Energy Conservation for Ad Hoc Routing," *Proc. Mobicom*, July 2001.
- [32] W. Yuan and K. Nahrstedt, "Buffering Approach for Energy Saving in Video Sensors," *Proc. IEEE Int'l Conf. Multimedia and Expo (ICME)*, 2003.
- [33] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proc. IEEE*, vol. 83, no. 10, Oct. 1995.
- [34] H.Y. Zhang, Y. Ge, C.J. Hou, and L. Sha, "Energy-Efficient Real-Time Scheduling in IEEE 802.11 Wireless LANs," *Proc. IEEE 23th IEEE Int'l Conf. Distributed Computing Systems (ICDCS)*, 2003.
- [35] H. Zhu and G. Cao, "A Power-Aware and QoS-Aware Service Model on Wireless Networks," *Proc. IEEE INFOCOM*, 2004.



Hao Zhu received the BS degree from Xian Jiaotong University, Xian, China, and the MS degree in computer science and engineering from the Institute of Software, Chinese Academy of Sciences, Beijing, China. He received the PhD degree from the Pennsylvania State University in 2004. His research interests include resource management, QoS, power efficient protocols, and medium access control protocols in wireless networks.



Guohong Cao received the BS degree from Xian Jiaotong University, Xian, China. He received the MS and PhD degrees in computer science from the Ohio State University in 1997 and 1999, respectively. He joined the Department of Computer Science and Engineering at the Pennsylvania State University in 1999, where he is currently an associate professor. His research interests are mobile computing, wireless networks, and distributed fault-tolerant computing.

His recent work has focused on data dissemination, cache management, network security, and resource management in wireless networks. He is an editor of *IEEE Transactions on Mobile Computing* and *IEEE Transactions on Wireless Communications*, has served as a cochair of the Workshop on Mobile Distributed Systems, and has served on the program committee of numerous conferences. He was a recipient of the Presidential Fellowship at the Ohio State University in 1999 and a recipient of the US National Science Foundation CAREER award in 2001.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.