# Energy-Efficient Sleep Scheduling for Delay-Constrained Applications Over WLANs

Lu Liu, *Student Member, IEEE*, Xianghui Cao, *Member, IEEE*, Yu Cheng, *Senior Member, IEEE*, and Zhisheng Niu, *Fellow, IEEE*

***Abstract*—In wireless local area networks (WLANs), switching of the nodes to low-power sleep mode to save energy is common. However, if sleep time is not properly scheduled, significant delays can occur, which is undesirable in delay-constrained applications. In this paper, we propose a novel energy-efficient sleep-scheduling protocol for maximizing the sleep time lengths of each node while satisfying the delay constraints. We first consider the single-user case and present the basic steps that the node takes to decide when and how long it can sleep. For the multiuser scenario, to mitigate channel contention in packet downloading after sleeping, the sleep schedules requested by the nodes are coordinated by the access point (AP) to avoid overlapping active epochs. Simulation results demonstrate that the proposed sleep-scheduling algorithm achieves lower loss rate while achieving higher energy efficiency than that of the existing GreenCall method.**

***Index Terms*—Delay-constrained applications, energy efficiency, sleep scheduling, wireless local area networks (WLANs).**

## I. Introduction

WIRELESS local area networks (WLANs) based on the IEEE 802.11 standard have been applied in many areas in which most of the applications are delay sensitive [1]. For example, current smartphones are normally equipped with WLAN interfaces that can support real-time multimedia data offloading [2]. WLANs also facilitate online calls through the voice over Internet protocol (VoIP) [3], [4]. These applications usually impose strict delay constraints on the data transmitted in the networks. Each packet needs to arrive at its destination before a prescribed deadline; otherwise, it will be dropped.

Another critical challenge in wireless networks is saving energy [5]–[9]. The IEEE 802.11 standard defines a few power-saving techniques for WLANs, including power-saving mode (PSM) [10] and automatic power-saving delivery (APSD) [11], which reduce energy consumption by switching the node from idle sensing mode to sleep mode. However, an energy-efficient design should also meet the network quality-of-service (QoS) requirements such as end-to-end delay [12]. If sleep time is not properly scheduled, significant delays can occur, which is undesirable in delay-constrained applications. Therefore, these power-saving techniques need to be enhanced to better accommodate sensitive delay constraints [5].

In this paper, we develop energy-saving techniques for delay-constrained applications over WLANs by dynamically switching a node to sleep mode, where our goal is to maximize the length of sleep time under packet deadline constraints. Take VoIP for example. Normally a VoIP packet can arrive at the destination ahead of its playout deadline. The GreenCall [13] algorithm takes advantage of this fact and puts the node into sleep mode according to the amount of spare time before the playout deadline. While sleeping, the downlink packets to the nodes are buffered at the access point (AP). When a node wakes up, it then retrieves the buffered packets from the AP and plays them out. The length of sleep time is calculated to ensure timely retrieving of the packets. To maximize energy savings, the length of the sleep period is to be chosen so that the packets are played out right before the deadline. With such an algorithm, a sleep/wake-up schedule can be computed that allows the node to remain in sleep mode for significant periods of time.

Primarily designed for the scenario with a single user in the WLAN, the GreenCall algorithm meets with challenges in multiuser scenarios. When a node wakes up and attempts to retrieve the buffered packets from the AP, the channel may not be available due to transmissions between the AP and other nodes. An extra delay will be incurred while waiting for the channel to become idle, which may consequently cause playout time violation and packet dropping. To solve this issue, our idea is to ensure that at any time there can be at most one node accessing the channel to retrieve buffered packets. To this end, each node can make a reservation with the AP in advance by sending a sleep request to the AP indicating that it would like to sleep now and will later occupy the channel after waking up. The AP checks whether there is any conflict with the schedules of other nodes and then decides to approve or decline the current sleep request. Once a request is approved, the node enters sleep mode; otherwise, it will stay active and send another request later. With this coordination mechanism, the active periods of different nodes are properly shifted to avoid collision, and a node can immediately start data retrieving after waking up.

Manuscript received August 24, 2013; revised January 20, 2014; accepted February 27, 2014. Date of publication March 21, 2014; date of current version June 12, 2014. This work was supported in part by the National Science Foundation under Grant CNS-1053777 and Grant CNS-1320736. The review of this paper was coordinated by Prof. Y. Fang.

L. Liu, X. Cao, and Y. Cheng are with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: xcao10@iit.edu).

Z. Niu is with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2014.2313114

0018-9545 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

The main contributions of this paper can be summarized as follows.

- We develop an energy-efficient algorithm for reducing the energy consumption of delay-constrained applications in WLANs by switching the nodes to a low-power sleep mode. The algorithm determines sleep period and wake-up time to maximize energy saving while guaranteeing packet delay constraints.
- To accommodate the multiuser scenario, a novel scheduling method is proposed. By exchanging sleep requests between nodes and the AP, downloading epochs for different nodes after waking up can be properly arranged for collision-free downlink data retrieving.
- Extensive simulation results are presented to demonstrate the energy savings achieved by the proposed algorithm over a wide range of network scenarios with different parameter settings. The results demonstrate that the proposed sleep scheduling achieves lower loss rate while having higher energy efficiency than the GreenCall method.

The remainder of this paper is organized as follows. Section II reviews related work. Section III presents the problem statements. Section IV develops the energy-efficient sleep-scheduling algorithms and Section V presents theoretical analysis of the performance. Simulation results are presented in Section VI. Section VII gives the concluding remarks.

## II. Related Work

The carrier sense multiple access with collision avoidance (CSMA/CA)-based IEEE 802.11 medium access control (MAC) protocol requires the nodes to keep sensing the channel or staying in idle mode when they are neither transmitting nor receiving, which consumes a significant portion of the energy resources of mobile nodes. Therefore, a promising strategy for power saving is to switch nodes when doing idle listening to sleep mode by turning off the wireless interfaces, thus saving a considerable amount of energy. With the IEEE 802.11 PSM technique, a node periodically enters sleep mode and wakes up to retrieve buffered downlink packets from or report uplink packets to the associated AP [5], [14].

However, since extra delay is introduced by the PSM, QoS becomes a challenging issue. Downlink packets suffer from both the buffering delay at AP and the delay due to channel contention of the destination nodes after waking up. To solve this, centralized scheduling methods at the AP, which decide the downlink sequence of all the nodes to either minimize contention intensity among the nodes [15] or minimize total waiting time and/or attain fairness [16], have been considered in the literature. Due to the fixed length of the sleep period, fluctuations of traffic rate will degrade the performance of PSM. Therefore, IEEE 802.11e provides the power-saving technique based on dynamic sleep period, i.e., APSD, in [17]. In the scheduled version of APSD, the service period (SP) for each node is determined by the AP's centralized scheduling. Lee and Hsieh propose an APSD-based algorithm in [18] to minimize the possible overlaps between SPs, which however is limited in general traffic without specifying the delay constraints. Aside from centralized solutions, unscheduled APSD is defined as a distributed method where no centralized AP scheduling is required and each node triggers the downlink process from AP by uplink packets. Each node can decide the trigger frequency by the arrival rate of its individual downlink packets and packet deadline [19]. As shown in [20], a node can turn to sleep mode in bad channel conditions and wake up to transmit packets when higher channel quality exists. Because of the distributed behaviors of nodes, channel contentions introduced by random channel access still impact the achieved energy efficiency. The authors in [21] discuss service differentiation in their work. They assign high priority to time-sensitive traffic and sacrifice the transmission delay of other traffic at relatively low cost by varying the sleep period. However, contentions within the same category of traffic are not fully solved. In this paper, the QoS requirement is achieved by strictly considering the deadline of each packet. The scattered channel access of each node is squeezed to bursts to mitigate contention. Although we keep the distributed manner of channel access, AP also participates in coordination to avoid overlapping of channel utilization from different nodes.

In VoIP networks, silence periods have been exploited to save energy. For example, for the large inactive periods during communication, optimal sleep window parameters are analyzed in [22] and the characteristics of human speech are used to conserve energy in [23]. However, these works only apply to relatively long silent periods during VoIP calls. Voice activity detection [24] and silence period prediction [25] are performed to reduce energy consumption, but the estimation introduces additional complexity or computation overhead. In contrast to these techniques that save energy only during silent periods, our algorithm can achieve energy conservation regardless of whether the client is silent or not.

Namboodiri and Gao propose the GreenCall algorithm with the information of single-packet transmission deadlines, where they use dynamic sleep period to accommodate delay constraints [13]. The length of sleep period is selected to make sure packets can arrive before playout deadline. Since the downlink rate from the AP to the mobile node is much larger than the packet arrival rate, the node only needs to stay awake for a short time period to retrieve downlink packets and use the spare time for sleep. The GreenCall algorithm works well for single-user case, but when there are multiple nodes, the active periods of different nodes may overlap and the subsequent contention will lead to additional delay, which causes the packets to miss their deadlines. To avoid such overlapping, Wang and Xu propose an uplink scheduling algorithm by arranging the sleep period of each user alternately [26]. The authors in [27] also use central scheduling to arrange transmissions. These centralized methods are not compatible with the GreenCall algorithm. In this paper, we propose an algorithm that can tackle this drawback while keeping the efficiency of GreenCall. When each node attempts to enter sleep mode, it will check whether the channel will be idle at the time it wakes up. This is inquired of the AP since it has the sleep/wake-up time information of all the nodes. With this method, the active periods of all the nodes are staggered and the deadline constraints can be better accommodated while energy saving can still be achieved.

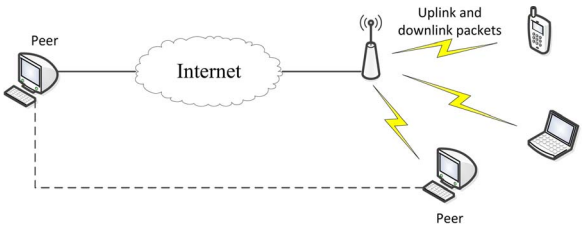



Fig. 1. Illustration of a VoIP system with WLAN.

## III. Problem Statement

Here, we present the energy-saving problem in delay-constrained applications over WLANs. For ease of exposition, we take VoIP over WLANs for example. As shown in Fig. 1, the sender and receiver (peers) of a VoIP call can access the Internet via wireline/wireless networks. WLAN can be used to bridge mobile VoIP nodes to the Internet, which is also advantageous over cellular Internet method in terms of low energy cost and extended coverage in indoor environments [13]. In the WLAN, the uplink and downlink communications for sending and receiving VoIP packets of the VoIP nodes are coordinated by an AP, based on the IEEE 802.11 protocol. Specifically, the CSMA/CA MAC protocol is applied to coordinate transmissions and avoid packet collisions. For ease of exposition, we describe in the following the receiving procedure of a VoIP packet since the sending part is symmetric. Upon receiving a packet, the AP will download it to the corresponding receiver node. Once the packet reaches the node, it will be stored in a playout buffer for a while to compress jitter in playback. For smooth playback, the final playout time of a packet should not exceed a specific deadline; otherwise, the packet is dropped. Packet drop rate is an important measurement of the VoIP performance and is mainly caused by large delay between the peers of a call. Apart from the Internet delay (i.e., delay introduced during transmission through the Internet), the MAC layer delay for the wireless communication between AP and the node, packet encoding and decoding delay, and delay in the playout buffer play vital roles.

Since energy is an important concern particularly for power-constrained mobile nodes, VoIP nodes with WLAN interfaces want to work with low power while matching the playout deadline requirements. The contention-based CSMA/CA protocol incurs a considerable amount of energy waste for a node in idle listening. The PSM provided in the IEEE 802.11 protocol allowing a node's radio to switch from active to sleep mode can save a significant amount of idle listening energy. With PSM, whenever a node wants to sleep, it should inform and get acknowledged by the AP, such that, during the node's sleep period, its downlink packets will be buffered at the AP for future transmission. The AP periodically broadcasts beacons containing traffic indication maps (TIMs) to indicate the buffer state for each node. On the other hand, the node frequently wakes up to check the beacons and prepares to download packets from the AP if the corresponding TIM indicates to do so; otherwise, if there is no packet buffered in the AP, the node will go back to sleep again.

PSM is suitable for VoIP communications in two aspects. Each packet has a certain level of delay tolerance; mobile nodes may have large portions of time without transmitting or receiving packets. Therefore, by turning the radio from the idle listening state to the sleep state, PSM is potentially capable of saving a large amount of energy while guaranteeing packet deadline.

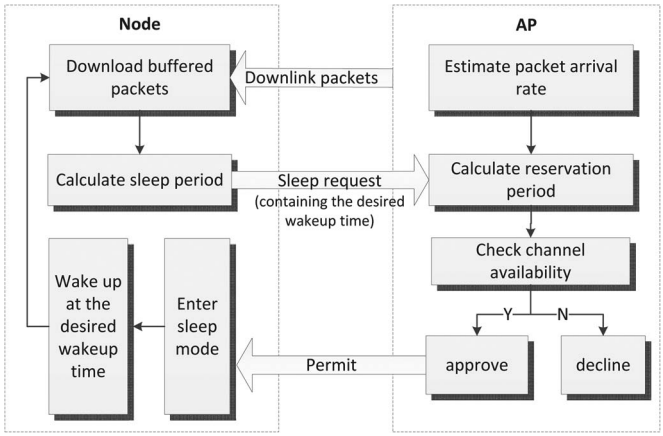



Fig. 2. Block diagram of system.

In view of the strict playout deadline requirement, the sleep periods of the nodes in WLAN must be appropriately scheduled. First, to save more energy, a node expects to sleep as long as possible. However, if its sleep period is excessively long, the downlink packets buffered in the AP during the node's sleep period may suffer from long delays and may fail to reach the node in time. Second, in the case with multiple nodes in the WLAN, if not appropriately coordinated, the active periods of nodes may introduce downloading conflict to each other. For example, when a node wakes up it may have to wait for a while (introducing more delay to its downlink packets) since the AP is busy handling another node's downloading. Finally, a node may have both downlink and uplink traffic, whose delays are affected by the sleep-scheduling method. In the next section, we design energy-efficient sleep-scheduling algorithms for VoIP systems, leveraging the PSM mechanism. We describe our algorithms in response to each of the aforementioned problems.

## IV. Energy-Efficient Sleep-Scheduling Algorithms

In our algorithms, each node attempts to sleep at the current time and wake up at some future time by sending request messages to the AP spontaneously. Only when the request is permitted by the AP will the node switch to sleep mode and wake up at the decided time (see Fig. 2). To explain the detailed operations at the node, we start with a simple scenario where there is only one node with downlink traffic served by the AP in the WLAN. The algorithm is then extended to multiuser cases and to accommodate uplink traffic.

### A. Single-User Scenario

Generally, a downlink packet arrives at the AP ahead of its playout deadline. In fact, this can be approximately achieved if the sender peer applies time control during the packet's uplink stage, as discussed later in Section IV-C. Therefore, if we slightly postpone a packet's arrival at the node, it may still

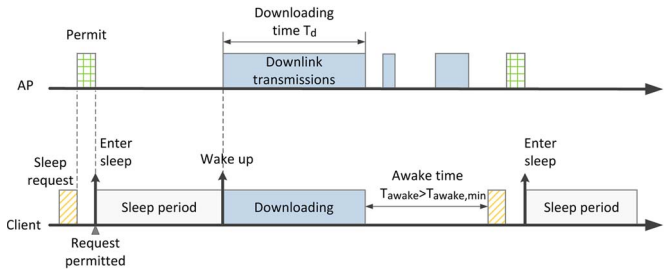



Fig. 3. Sleep period scheduling in the single-user scenario.

be able to catch up the deadline. Taking advantage of this point, the node can switch to sleep mode for some time and wake up before the deadline to retrieve the packet from the AP and play it out.

*1) Algorithm:* In PSM, during the sleep period, a node should frequently wake up for a short period to check whether the AP has buffered packets to be downloaded. In our case, since each packet has delay tolerance, the node can sleep continuously without wasting energy in frequent mode changing. As shown in Fig. 3, our algorithm runs as follows. Suppose that the node has received $M$ downlink nonbuffered packets continuously when it is in active mode. Since sleep will introduce additional delay to the packets, we first need to figure out how much extra delay one packet can tolerate without missing its deadline. Upon reception at the node, each packet has delay tolerance, denoted as $D_n$, which is the tolerable amount of time before being played out excluding the constant time $T_{\rm dec}$ spent in packet decoding at the node. In other words, the packet can only tolerate an additional period of delay $D_n$ to be played out in time. Let $D_{n,\min}$ be the minimum delay tolerance of those $M$ packets. If the value of $D_{n,\min}$ is large enough, the node can consider switching to sleep mode. Specifically, if $D_{n,\min} > T_{s,\min}$, the node will decide to sleep. $T_{s,\min}$, the minimum sleep period, is used to prevent each node from frequently switching between active and sleep modes, which can cause significant amounts of energy consumption to the node [28]. Meanwhile, a large $T_{s,\min}$ may become undesirable since nodes may seldom enter sleep mode in this case, thus opposing the saving of energy. In our algorithm, $T_{s,\min}$ is set as 500 ms, i.e., a moderate value that each node has a good chance to sleep upon receiving a fresh packet while preventing frequent mode switching.

Once sleep switching is decided, the node determines a sleep period $T_s$ (which is discussed later in Section IV-A2) and sends the sleep ending time (i.e., the next wake-up time) via a *Request* message to the AP. The latter then checks the future channel condition and confirms the request by replying with a *Permit* signal. However, the AP can reject the request by replying nothing. The node switches to sleep immediately after receiving the permit and wakes up at the determined time. Otherwise, if no permit is received after a period of $T_{\rm wait}$, it remains in active mode and generates a new request later. All the downlink packets that arrive during the sleep period will be buffered at the AP and will be transmitted to the node after the wake-up time. After retrieving all buffered packets, the node must stay in active mode for at least a minimum awake time $T_{\rm awake,min}$ to be able to receive a number of packets and update the minimum delay tolerance $D_{n,\min}$. The packets received after the reserved

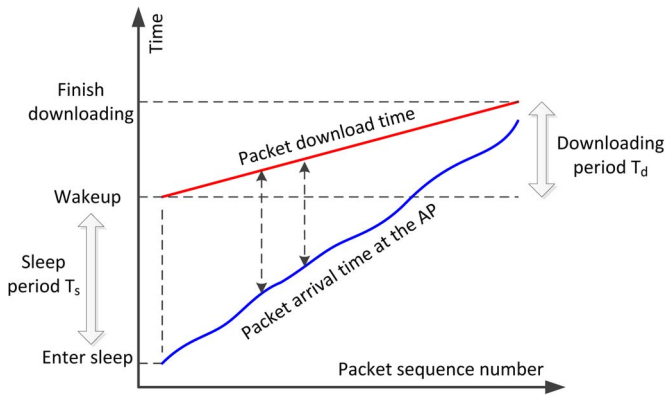



Fig. 4. Timing of packet arrival and downloading of a node. (The gap between the two curves indicates the packet buffering at the AP plus downloading delay. Notice that the packet arrival time in Fig. 4 is presented as a generic increasing curve to describe a general packet arrival process.)

downloading period have not been buffered by the AP and therefore have relatively long delay tolerance. If we do not stay active to wait for the nonbuffered packets, the delay tolerance will be derived from the buffered packets, which will be much smaller since the buffering time will reduce delay tolerance (i.e., delay tolerance depends on when the packet arrives at the node). In this case, the delay tolerance may be too short to trigger the sleep request. As a result, the node will wait for additional time and update the delay tolerance for the next sleep attempt. Although our scheme can also work in this case, there are some unnecessary computations. Therefore, we set the node to remain active to get nonbuffered packets, which can provide long-enough delay tolerance to trigger sleep request.

Instead of staying active all the time for scattered packet arrivals, this method compresses the packet arrivals to a short period for the node's downloading and the spare time is used for sleeping. In addition, the sleep time is chosen to guarantee that packets arrive before playout deadlines. In this way, idle listening time is greatly reduced and energy efficiency can be improved without sacrificing network performance.

*2) Deciding the Sleep Period:* The length of sleep period should be carefully designed to guarantee that each packet can arrive at the node in time. Once the node switches to sleep mode, all its downlink packets during sleep time will be buffered at the AP, which will introduce additional delay to these packets. Among these buffered packets, the packet with maximum extra delay (denoted as $D_{b,\max}$) is the bottleneck, where $D_{b,\max}$ consists of the time spent in being buffered at the AP and retrieved by the node. In other words, the buffered packets suffer at most $D_{b,\max}$ length of extra delay caused by sleep scheduling. However, we are not able to obtain the actual values of the buffered packets' delays before they arrive at the node. A safe and reasonable strategy is to use the delay of the first buffered packet as $D_{b,\max}$ since it suffers the longest buffering delay and has the earliest playout deadline. As long as this packet can arrive at the node before its deadline, the playout deadlines of all the other buffered packets can also be caught. As shown in Fig. 4, since the packet downloading rate is faster than the arrival rate,[1] the delay will decrease along time

[1]Otherwise, the buffer size of the AP will be unstable.

during the sleep period. The buffered time of the first packet is the same as the length of sleep period. Therefore, the maximal length of time the node can sleep is

$$T_{s,\max} = D_{b,\max} - \frac{L}{r_d} \tag{1}$$

where $r_d$ is the downloading rate, and $L$ is the size of a packet.

In the equation, the maximum delay $D_{b,\max}$ is still unknown to the node when deciding the sleep period. We instead use an estimation $\hat{T}_{s,\max}$ as the maximum sleep length based on the time information of previously received packets. Since the extra delay $D_{b,\max}$ cannot exceed the delay tolerance, we use $D_{n,\min}$ as $D_{b,\max}$ since the former is the bound of the latter. Here, the value of $D_{n,\min}$ is obtained by the minimal delay tolerance of the nonbuffered packets, which are received after the last downloading period. Since the node must stay in active mode for at least $T_{\text{awake,min}}$, $D_{n,\min}$ can be determined based on the packets received within this period. In the rare case that there is no packet received during the awake time, $D_{n,\min}$ calculated in the previous sleep decision will be used. Then, the maximum sleep period is determined as

$$\hat{T}_{s,\max} = D_{n,\min} - \frac{L}{r_d}. \tag{2}$$

When the network delay fluctuates widely, the value of $D_{n,\min}$ may be overestimated, resulting in an inaccurate estimation of sleep length, which may lead to delay violation. In a conservative way, we do not directly take $\hat{T}_{s,\max}$ as the length of sleep period. Instead, we deduct $\hat{T}_{s,\max}$ by an amount of time, i.e., the *sleep guard* (denoted as $T_{s,\text{guard}}$), and set the remaining value as the actual length of sleep period, which is

$$T_s = \hat{T}_{s,\max} - T_{s,\text{guard}}. \tag{3}$$

$T_{s,\text{guard}}$ can be set according to the fluctuation of network delay. For example, if the Internet delay is 100 ms $\pm$ 10 ms, the value of $T_{s,\text{guard}}$ can be set to 10 ms. Adjusting this value provides the tradeoff between energy saving and packet loss.

As shown in Fig. 4, when all the buffered packets have been retrieved, the packet arrival and downloading curves meet if there are no packets that arrived after downloading began. Otherwise, there may still be a gap between two curves, which represents the buffering delay of the latest buffered packet. Once downloading is completed, the node can prepare to calculate the next $T_s$ and switch to sleep mode again. The node will repeat the sleep/wake-up process periodically to reduce idle listening time and save energy.

*3) Estimating the Downloading Period:* Although in the single-user scenario the AP does not need to calculate the length of downloading period when deciding whether to confirm a request, we present the estimation of the downloading period here as a preliminary for the algorithm running in the multiuser scenario. As shown in Fig. 4, denote $\tilde{r}_a$ as the packet arrival rate averaged over the sleep period $T_s$. Note that $\tilde{r}_a$ may be time varying and its long-term average is denoted as $r_a$. After the node wakes up, it starts downloading all the buffered packets from AP in a first-in–first-out order, which is described by the packet download timeline in the figure. Denote the length of time spent in retrieving buffered packets as $T_d$. It should be noticed that during downloading of buffered packets, there may also be packets arriving at the AP. These packets will be buffered shortly and then sent to the node following the previously buffered packets. Therefore, we have the following relationship:

$$(T_s + T_d)\tilde{r}_a = T_d r_d \tag{4}$$

where $r_d$ is the downloading rate from AP to the node, which is reversely proportional to the slope of the packet downloading line in the figure. Since $\tilde{r}_a$ is unknown to the AP beforehand, its long-term average $r_a$ is used as an estimate, where we assume that $r_a$ is a known parameter. In this sense, the estimated downloading period $\hat{T}_d$ satisfies

$$(T_s + \hat{T}_d)r_a = \hat{T}_d r_d. \tag{5}$$

### B. Multiuser Scenario

In the multiuser scenario, the challenge is that when a node wakes up and wants to retrieve the buffered packets, the AP may not be available or the channel may be busy due to other user's transmissions. Additional delay will be introduced when waiting for the channel to be idle, which may result in violation of deadlines and packet drop. In addition, if the channel is occupied by another node, which is also running this algorithm, the delay will be much longer since the buffered packets are continuously retrieved and the channel will be occupied during the entire process of downloading. To solve this issue, the idea is to ensure that at any time there is at most one node using the channel to retrieve buffered packets so that the consequent data downloading will not collide with other transmissions. If one node can predict whether the channel will be occupied later when it wakes up, it can decide whether to sleep based on the prediction. If the channel is idle when the node wakes up, it can switch to sleep at the present time; otherwise, the node will stay active and try to sleep later to avoid future contention. This requires knowledge of the wake-up time of all the nodes. In our algorithm, we use the AP to collect the required information and shake hands with each node informing it whether it should sleep. Since the AP knows all the reserved downloading periods of the nodes, it can easily check for a foreseeable future time when the channel will be idle (otherwise, there will be one node busy downloading data from the AP). Therefore, once receiving a sleep request from a node, the AP checks whether the channel will be idle at the proposed wake-up time of the node. If yes, the request will be approved. Otherwise, the request is declined since at the proposed wake-up time, another node (which has already reserved a future downloading period at the AP) will be downloading data from the AP. Most of the computation is done at the AP and communication overhead is minimized to avoid additional energy consumption at the nodes.

Our algorithm in the multiuser scenario is illustrated in Fig. 5. For ease of exposition, we consider two nodes, namely, node $i$ and $j$. A node decides whether and how long to sleep based on the same strategy previously described in the single-user scenario. Once it has decided to sleep, the node will send the request message encapsulating the wake-up time to the AP using the contention-based IEEE 802.11 CSMA/CA

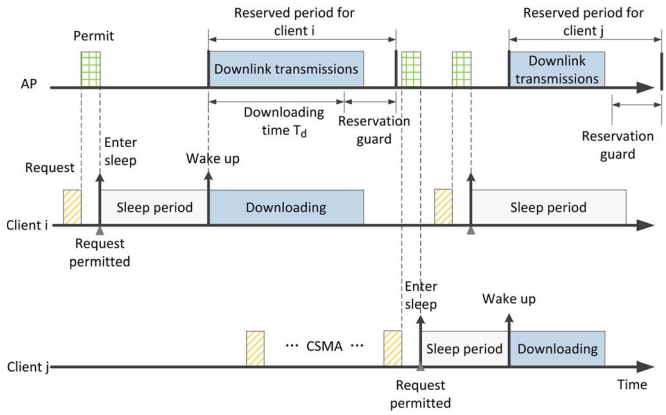



Fig. 5. Sleep period scheduling in the multiuser scenario.

protocol to avoid collisions with other nodes' transmissions. Upon receiving the request, the AP estimates the downloading period $\hat{T}_d$ according to (5) and attempts to reserve a period of $\hat{T}_d + T_{d,\mathrm{guard}}$ for the requesting node at its proposed wake-up time. $T_{d,\mathrm{guard}}$ represents a *reservation guard time* immediately after $\hat{T}_d$. Since there could be bursts of downlink packets arriving at the AP when the node is sleeping or busy downloading buffered packets, which cannot be predicted by the AP in estimating $\hat{T}_d$, the guard time can make room to transmit those packets. Under a constant or small-fluctuation arrival rate or when the downloading rate is high enough to accommodate an unpredicted burst, $T_{d,\mathrm{guard}}$ can be set to a relatively small value. Otherwise, it should be increased accordingly. In our simulations, since the arrival rate is constant and the downloading rate is relatively high, a short $T_{d,\mathrm{guard}}$, i.e., 5 ms, is selected. If the calculated reservation period does not overlap with any other served periods, the AP will mark the reservation for this node and confirm the request by replying the node with a permit message. Otherwise, the request is denied and AP will not reply to the node. In this case, the node will retry sending the request after waiting for a period of time. During the reserved period, the AP transmits buffered downlink packets to the designated node without collisions and buffers the downlink packets of the other nodes. Note that the reservation guard time is only used by the AP in the decision process and the actual downloading period may be shorter than the reservation. After the node has retrieved all buffered packets from AP in the downloading period, both AP and the node will return to normal active mode immediately (terminating the reservation) such that any downlink packet can be transmitted, and new request/permit messages can be also exchanged. As previously mentioned, after the downloading period, the node must stay in active mode for at least $T_{\mathrm{awake,min}}$ to receive nonbuffered packets.

In the case that the sleep request is lost and not received by AP, the node will not receive the permit message after $T_{\mathrm{wait}}$ and it can start a new request. Therefore, the packet loss of sleep request will not affect the normal working of this algorithm. If the permit message is lost, the AP makes a reservation, which is in fact invalid. The node will treat its request as rejected and send a new request. Then, on the AP side, it will receive a request from a node that is thought to be sleeping. Therefore, the AP can be aware of the loss of permit message then cancel the invalid reservation and recalculate the schedule according to the new request. From the analysis, it can be seen that the proposed algorithm is robust to the loss of request/permit packets.

In the proposed algorithm, although the AP participates in the coordination, the sleep time of each node is determined by the node in a distributed manner. Regarding fairness, it can be ensured in view of the following aspects. First, since the length of reservation time is decided by the AP, it can decline a request if the corresponding $T_d$ is too large, i.e., to prevent some node occupying channel for too long, thus ensuring the access probabilities of all the nodes. Second, each time a node successfully makes a reservation, it will then go to sleep mode, and during this time, the other nodes can apply for sleep and reservation. Meanwhile, during the consequent awake time of a node after its reservation period, other nodes can shake hands with the AP for sleeping and reserving a designated period in the future. In this way, the AP can be prevented from allocating reservation periods for some node consecutively.

Similar to the single-user scenario, the algorithm converges the scattered downlink packets into bursts during the downloading period. In the multiuser scenario, the algorithm can also mitigate the channel contention since most of the packets are transmitted in reserved periods.

### C. Dealing With Uplink Traffic

Since our algorithm is focusing on guaranteeing the delay of received packets, we mainly discuss downlink traffic in the previous content. The proposed scheduling method is based on the playout deadline of a packet, which means that the scheme is performed at the downlink side. However, the scheme is compatible with uplink traffic. Basically, a node will attempt to send out an uplink packet as soon as the packet is generated. If an uplink packet is generated during sleep period, it will break the sleep state and bring the node to active mode, which will simply reduce the sleep length but will not disturb the proposed algorithm. If the uplink packet is generated during reservation period, it can be sent after downloading. A small amount of delay is introduced in this case, but it is tolerable since the reservation period is generally short.

If uplink packets can be buffered at the node during sleep mode, the sleep period can be preserved and the buffered uplink packets can be sent along with retrieving downlink packets during the reservation period. In this case, the node can wake up ahead of the calculated time to send out uplink packets. Otherwise, the node sends out the uplink packets immediately after retrieving the downlink packets during reservation period, and correspondingly, we lengthen the calculated $T_d$ to cover the uploading time. In this case, delay of uplink packets is introduced due to the sleep period. If both peers of a VoIP call are using the proposed algorithm, for fairness, each can use part of the calculated $T_s$ (e.g., 50% each) as the actual sleep time to keep the delay within constraint.

If the uplink rate is much larger than the downlink rate, then it may not be necessary to apply the proposed algorithm since the actual sleep period may be too short. For example, in a VoIP conversation, the algorithm can be triggered when the node is listening or in mutual silence period.

## V. Performance Analysis

Here, we derive analytical models of the system performance in terms of energy efficiency. Particularly, we focus on the percentage of energy-efficiency improvement, denoted as $\eta$, as compared with and without applying the proposed algorithms. For simplicity, we only consider downlink in our analysis. We assume that the energy consumption due to mode switching is neglectable and that the Internet delay of downlink packets when arriving at the AP follows a continuous distribution with cumulative distribution function $F(x)$.

### A. Single-User Scenario

In the single-user scenario, the node applying the algorithm periodically alternates between sleep mode and downloading packets. Denote $P_t$, $P_i$, and $P_s$ as the power consumption in transmitting/receiving, idle sensing, and sleep mode, respectively. Taking a generic sleep/wake-up period for consideration, the expected total energy consumption of one node is

$$E = \mathbb{E}[T_s]P_s + (\mathbb{E}[T_d] + T_{RP})\,P_t + T_{\text{awake}}P_i \tag{6}$$

where $T_{RP} = T_{\text{request}} + T_{\text{permit}}$ with $T_{\text{request}}$ and $T_{\text{permit}}$ as the times for sending a request and receiving a permit, respectively. $T_{\text{awake}} \geq T_{\text{awake,min}}$ is the mandatory awake time during which the node must stay in active mode. To maximize energy saving, the node will not spend unnecessary time in active state; therefore, $T_{\text{awake}} = T_{\text{awake,min}}$.

The energy consumption during the same length of time if not applying our algorithm is

$$E_0 = (\mathbb{E}[T_d] + T_{RP})\,P_t + (\mathbb{E}[T_s] + T_{\text{awake}})\,P_i. \tag{7}$$

In the equation, the expected sleep period $\mathbb{E}[T_s]$ and downloading period $\mathbb{E}[T_d]$ are to be derived. To obtain $\mathbb{E}[T_s]$, we need to find the long-term average of $D_{n,\min}$, as shown in (2). Since $D_n$ is calculated by subtracting $T_{\text{dec}}$ and network delay from the corresponding deadline where both $T_{\text{dec}}$ and the deadline are constant, we can easily get the distribution of $D_n$ based on the distribution of network delay. Suppose the distribution function is $F(x) = \mathbb{P}[D_n < x]$ with probability density function $f(x)$ such that $\int_0^x f(x) = F(x)$. The function can be obtained by analyzing the arriving time of historical packets. Then, the cumulative distribution function of $D_{n,\min}$ can be calculated as

$$\begin{aligned} F_{\min}(x) &= \mathbb{P}[D_{n,\min} < x] \\ &\approx 1 - (1 - \mathbb{P}[D_n < x])^M \\ &= 1 - (1 - F(x))^M \end{aligned} \tag{8}$$

where we treat the delay tolerances of the $M$ packets independently, whereas practical delays of adjacent packets may be correlated to some extent and hence use approximating in the second line. With the previous equation, the expected value of $D_{n,\min}$ is

$$\mathbb{E}[D_{n,\min}] \approx \int_0^\infty x \frac{d}{dx} F_{\min}(x). \tag{9}$$

Taking expectations at both sides of (2) and (3) then substituting (9), we can obtain $\mathbb{E}[T_s]$. According to (5), we have $\mathbb{E}[T_d] = (r_a/r_d - r_a)\mathbb{E}[T_s]$. Then, since the numbers of received packets for these two cases are the same, the energy efficiency can be improved by a factor

$$\begin{aligned} \eta &= \frac{E_0}{E} - 1 \\ &= \frac{\mathbb{E}[T_s](P_i - P_s)}{\mathbb{E}[T_s]P_s + (\mathbb{E}[T_d] + T_{RP})\,P_t + T_{\text{awake}}P_i}. \end{aligned} \tag{10}$$

### B. Multiuser Scenario

We first consider a homogeneous network where all the nodes are identical and their downlink traffic follows the same distribution with average arrival rate $r_a$. For each node, since its sleep periods and downloading periods are determined based on the arrival information of its own nonbuffered packets, the average lengths of sleep periods $\mathbb{E}[T_s]$ and downloading periods $\mathbb{E}[T_d]$ are the same in the single-user scenario. Moreover, the long-run averages of the two periods are also the same for all nodes under the homogeneous network assumption. Because the algorithm runs fairly for each node, in the ideal case, the timeline of the AP will show a pattern that the nodes occupy reservation periods alternately and the average number of reservation periods for the nodes should be the same. Suppose there are $n$ nodes. Note that during the downloading period, the AP only serves the reserved node, ignoring the sleep requests from others. In this sense, the other nodes cannot occupy the channel during downloading period, and there will be no overlapping between downloading and request/permit transmissions. Therefore, for each node, the average length of time it is occupying the channel in one sleep/wake-up round can be approximately evaluated as $\Delta = T_{RP} + \mathbb{E}[T_d] + ML/r_d$, where $ML/r_d$ accounts for the time that the client receives averagely $M$ nonbuffered packets during the awake period. Now, consider a total of $n\Delta$ time in which the clients take turns to sleep, we must have

$$n\Delta \leq T_{RP} + \mathbb{E}[T_d] + \mathbb{E}[T_s] + T_{\text{awake}} \tag{11}$$

$$n\Delta r_a \leq \mathbb{E}[T_d]r_d + ML \tag{12}$$

where the first line shows that the sleep and awake periods of a node should be long enough to accommodate other node's downloading periods; otherwise, a certain amount of packets will be dropped at the AP due to limited channel capacity, no matter whether the proposed algorithm is applied or not. The second line asserts that the packet downloading rate matches the arrival rate so that the network is stable. Moreover, (11) can be reduced to

$$T_{\text{awake}} \geq (n-1)\,(T_{RP} + \mathbb{E}[T_d]) - \mathbb{E}[T_s] + n\frac{ML}{r_d} \tag{13}$$

which gives a lower bound for the awake period. Therefore, we assume that both (11) and (12) are satisfied in the following.

For each node, the energy-efficiency improvement can be calculated following the same way as previously presented in (6)–(10) except that the awake time now may be longer. In view of (13), $T_{\text{awake}}$ can be set as $\max\{T_{\text{awake,min}}, (n-1)(T_{RP} + \mathbb{E}[T_d]) - \mathbb{E}[T_s] + n(ML/r_d)\}$.

TABLE I
DEFAULT SIMULATION PARAMETER SETTINGS

| | |
|---|---|
| Number of mobile nodes $n$ | 3 |
| VoIP packet size $L$ | 160bits |
| Request/permit packet size | 20bits |
| Downloading rate $r_d$ | 1000pkt/s |
| Packet lifetime | 1100ms |
| Internet delay | 100ms $\pm$ 10ms |
| Average packet inter-arrival time | 20ms |
| Minimum sleep period $T_{s,\min}$ | 500ms |
| Sleep guard $T_{s,guard}$ | 10ms |
| Reservation guard $T_{d,guard}$ | 5ms |
| Wait time $T_{wait}$ | 50ms |
| Minimum awake time $T_{awake,\min}$ | 50ms |
| Transmission/reception power $P_t$ | 787mW |
| Idle listening power $P_i$ | 503mW |
| Sleep mode power $P_s$ | 44mW |

In practice, since the nodes may be heterogeneous, $T_s$ and $T_d$ of each node may be different and varying along time; the distribution of reservation periods is much more random. In this case, it is possible that there are some users failing to obtain sleep chance at some time and remaining longer time in active mode. Therefore, the energy saving is less than the ideal value. However, since most of the sleep requests can be approved, the energy efficiency can be still improved a lot compared with not using the algorithm. We will demonstrate the effect of heterogeneity with simulations in the next section.

## VI. SIMULATION RESULTS

Here, we present simulation results of the proposed algorithm. We develop an NS2 [29] program to simulate a VoIP system over WLAN where the WLAN contains one AP and three VoIP mobile nodes. Parameter settings in our simulations are listed in Table I. In the proposed scheduling algorithm, each node calculates the length of next sleep period according to the delay tolerance of received nonbuffered packets, i.e., the time between original packet arrival at the AP and the playout deadline. We assume that the downlink packets are generated by the peers of the current receiving nodes every 20 ms, where they are assumed of the same length $L = 160$ bits. Before arriving at the AP, each packet experiences the Internet delay, which is modeled as a random value following uniform distribution in the range 100 ms $\pm$ 10 ms. In this case, the average packet arrival rate at the AP is $r_a = 50$ pkt/s. The *packet lifetime*, i.e., the time gap between packet generation and playout deadline, is used to calculate the delay tolerance. The minimum sleep period $T_{s,\min}$ is set to 500 ms. The sleep and reservation guard times, i.e., $T_{s,\text{guard}}$ and $T_{d,\text{guard}}$, are set to 10 and 5 ms, respectively. The time period $T_{\text{wait}}$ that a node should wait after sending a request message to receive the corresponding permit message from the AP is set to 50 ms. The minimum awake time $T_{\text{awake},\min}$ is configured to 50 ms. In the following, by default, we focus on the scenarios with only downlink traffics and evaluate the effect of uplink traffics later here.

### A. Performance Evaluation

The performance of our algorithm is measured by the percentage of energy saving $\eta$ (compared with the scenario without node sleeping) and the packet loss rate of each node. The value of $\eta$ derived from (10) is shown in some of the figures as the theoretical upper bound. The energy saving mainly depends on the portion of time that a node is sleeping, and the packet loss occurs if a packet misses its playout deadline or a collision happens. Generally, excessive Internet delay will lead to packet loss no matter whether our or other algorithms are used or not. To clearly demonstrate the performance of our algorithm, we focus on the packet loss incurred by the algorithm due to inaccurate estimation of the sleep period or collisions. Therefore, in our simulations, the Internet delay is set to be smaller than the packet lifetime (the time from packet generation to playout deadline) such that packet losses caused by Internet delay are excluded.

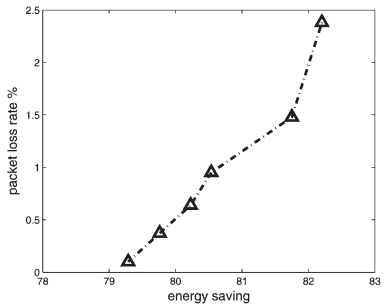



Fig. 6. Tradeoff between energy saving and packet loss rate with tuning of the sleep guard time.

*1) Tradeoff Between Energy Saving and Packet Loss:* Generally, the longer a node sleeps, the more likely its downlink packets miss the playout deadlines and hence the higher loss rate. Our simulation results show that, on average, our method can achieve 80% energy saving with packet loss rate around 1%. In the following, we present detail performance evaluations under various simulation scenarios. We first vary the sleep guard to evaluate the tradeoff between energy saving and packet loss rate.

As shown in Fig. 6, when we reduce the sleep guard time, both energy saving and packet loss rate increase. With a relatively short sleep guard, the nodes can stay in sleep mode for a longer time, as shown in (3), which may result in underestimated Internet delay. Using this inaccurate value to calculate the sleep period, the nodes may wake up and find some packets already missing the deadlines. A more conservative way to decide the sleep length is to use a larger sleep guard, which can greatly reduce the packet loss rate at the cost of lower energy saving.

*2) Impact of Internet Delay:* Fig. 7 shows the system performance under varying Internet delays. As the Internet delay increases, there is less time for sleeping and the node will wake up more frequently to retrieve packets buffered at the AP. As a result, the retrieved packets are less likely to miss the playout deadline and the loss rate will decrease. In this case, both the energy saving and the packet loss rate are reduced. If the Internet delay is very large the spare time is not long enough to initiate a sleep period, the sleep time will be further reduced.

We also compare the achieved energy saving with the theoretical upper bound (obtained based on the model proposed in Section V). The upper bounds are calculated assuming the ideal situation that all the sleep requests can be granted. In practice,

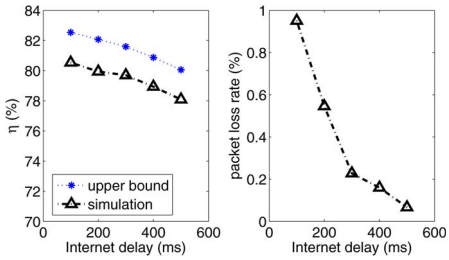



Fig. 7. Performance under different Internet delays.

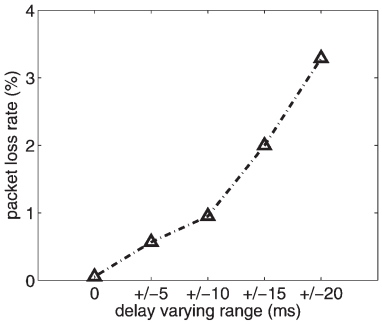



Fig. 8. Packet loss rate under different Internet delay varying ranges.

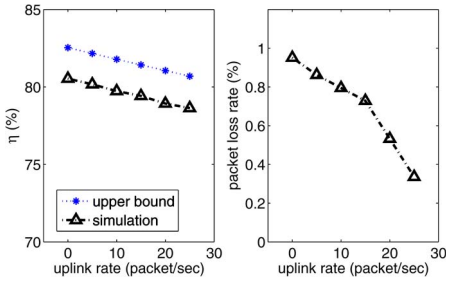



Fig. 9. Performance under different uplink traffic rates.

some sleep requests may be declined due to reservation conflict and the nodes have to wait for some time for next request. Therefore, the achieved energy saving from simulation results is slightly lower than the upper bounds. However, the difference is only around 2% so that the accuracy of our model can be verified.

The variance of Internet delay or equivalently the variance of packet interarrival time also has influence on the performance. Although it does not affect the sleep time percentage or energy saving due to averaging effect, large variance will cause a higher packet loss rate, as shown in Fig. 8. When the Internet delay is changing widely, the possibility that the delay is underestimated by the nodes becomes larger, which results in improper decision of sleep length and packet drop. To tackle this problem, we can conservatively choose a larger sleep guard (as discussed earlier).

*3) Impact of Uplink Traffic:* Fig. 9 shows the performance under different uplink packet generation rates. The uplink packets of a node will be transmitted after the downlink packet retrieving process. Therefore, to make room for uplink traffic, the reservation length is lengthened. Longer reservation time leads to smaller percentage of sleep time; hence, the energy savings decrease as the uplink traffic rate increases. In addition, staying active for a longer time period reduces the chance of missing packet deadline, and the packet loss rate is slightly decreased.

TABLE II
PERFORMANCE OF HETEROGENEOUS USERS

| settings | | energy saving | loss rate |
|---|---|---|---|
| Inter-arrival time (ms) | 20 20 20 50 50 | 79.01% | 0.65% |
| | 20 30 35 40 40 | 77.94% | 0.28% |
| Internet delay (ms) | 100 100 100 500 500 | 78.84% | 2.14% |
| | 100 200 300 400 500 | 78.16% | 1.58% |

*4) Heterogeneous Users and Fairness:* Previously, we assumed that all the nodes are homogeneous with the same settings. Here, we evaluate the performance with heterogeneous nodes, where five nodes with different packet arrival rates or Internet delays are used. Table II lists the simulation results.

In both homogeneous and heterogeneous situations, the average performance of each node is almost the same with that of the others, ensuring fairness among users.

*5) Overhead Analysis:* In the proposed algorithm, each node only needs to record the arrival time of the previous packet and perform a simple calculation to get the sleep request; hence, the computation overhead at the user side is negligible. As for the AP, the additional computation is to calculate reservation length and check its timeline, which are also simple manipulations. Therefore, our method has no computation complexity and very low computation overhead.

Communication overhead is introduced by sending sleep requests and receiving permit messages. As shown in Table III, the number of sleep requests is always larger than the theoretical value since there are rejected requests in practice. Similarly, the number of sleep/wake-up periods is slightly smaller than the ideal case, and therefore, the number of sleep permits (indicating the number of successful sleep/wake-up periods) is smaller than theoretical value. Finally, the communication overhead is about 1.3% compared with the normal data packets, which will not cause a large amount of additional traffic in the network.

## B. Performance Comparison With GreenCall

Since the major improvement offered by our algorithm compared with the GreenCall algorithm is the performance in multiuser scenarios, Figs. 10 and 11 present the performance comparison at different numbers of nodes. The sleep guard is set to 1 and 10 ms in the two figures, respectively.

In Figs. 10 and 11, we can observe that the energy saving of the GreenCall algorithm is slightly higher than that of our algorithm since there is no reservation arrangement and each node follows its own scheduling regardless of the potential conflict with the others. Contention during the retrieving process will incur additional waiting; hence, the energy saving is decreased as the number of nodes increases. In our algorithm, we need to negotiate sleep scheduling between nodes and the AP; therefore, the rejection of a sleep request will introduce some additional active time to the nodes. As the number of nodes increases, the sleep request is more likely to be declined; hence, there is a larger gap between upper bounds and our method. However, the gap is at most 4%, which is very slight.

TABLE III
COMMUNICATION OVERHEAD

| Minimum awake time | energy saving | number of request/permit per 100 packets | communication overhead |
|---|---|---|---|
| 50ms | 80.54% (82.54%)[a] | 5.58/5.42 (5.49) | 1.38 (1.37) |
| 100ms | 77.43% (78.93%) | 5.38/5.20 (5.24) | 1.32 (1.31)% |
| 150ms | 73.76% (75.63%) | 5.13/4.98 (5.02) | 1.26 (1.25)% |

[a]The numbers in parentheses are theoretical results.

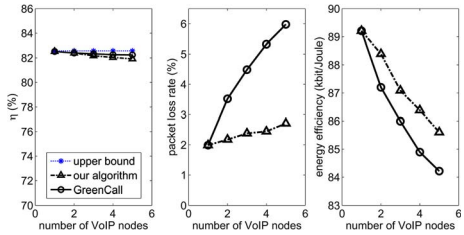



Fig. 10. Performance comparison (small sleep guard).

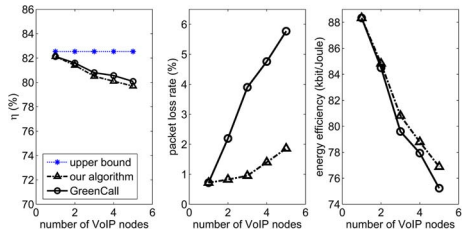



Fig. 11. Performance comparison (large sleep guard).

Although both algorithms suffer additional waiting time in the multiuser scenarios, the waiting turns into delay in the GreenCall algorithm but turns into normal active state in our algorithm. Consequently, our method has an advantage on the lower packet loss rate, which is demonstrated in the figures. As discussed in the previous sections, our method can avoid conflict among busy downloading periods of different nodes. Therefore, the contention can be mitigated, and the packet loss rate can be much lower than that of the GreenCall algorithm, which suffers more packet losses, particularly in multiuser scenarios. In both methods, the packet loss rate will increase with the number of nodes, but by the results of our method, the increase is not very fast compared with that of the GreenCall algorithm.

We further compare the energy efficiency of the two algorithms where energy efficiency is defined as the amount of data bits transmitted per unit of energy consumption. The results in Figs. 10 and 11 show that our algorithm outperforms the GreenCall algorithm in terms of energy efficiency in all cases. In summary, the algorithm proposed in this paper can achieve lower packet loss rate with only slight degradation of energy saving and can also achieve higher energy efficiency. The performance of our method is also more robust as the number of nodes increases.

## VII. CONCLUSION

In this paper, we have proposed an energy-efficient sleep-scheduling algorithm for delay-constrained applications over WLAN. Each node calculates its sleep period under packet delay constraint and requests that the AP reserve a future period for exclusive transmission. The AP examines the scheduling of all the nodes and makes sure that there are no overlapping reservation periods. In this way, the active periods of different nodes are staggered to mitigate contention and prevent further packet losses. The performance of the proposed algorithm in terms of energy saving and packet loss rate is evaluated by NS2 simulations and the results show that the algorithm can achieve significant energy saving while keeping the packet loss rate at a very low level. Comparison between our method and the existing GreenCall algorithm shows that the former outperforms the latter in terms of energy efficiency.

## REFERENCES

[1] M. Schaar, Y. Andreopoulos, and Z. Hu, "Optimized scalable video streaming over IEEE 802.11 a/e HCCA wireless networks under delay constraints," *IEEE Trans. Mobile Comput.*, vol. 5, no. 6, pp. 755–768, Jun. 2006.

[2] B. Han, P. Hui, V. S. A. Kumar, M. Marathe, J. Shao, and A. Srinivasan, "Mobile data offloading through opportunistic communications and social participation," *IEEE Trans. Mobile Comput.*, vol. 11, no. 5, pp. 821–834, May 2012.

[3] J. Tang and Y. Cheng, "Quick detection of stealthy SIP flooding attacks in VoIP networks," in *Proc. IEEE ICC*, Kyoto, Japan, Jun. 5–9, 2011, pp. 1–5.

[4] B. Goode, "Voice over Internet protocol (VoIP)," *Proc. IEEE*, vol. 90, no. 9, pp. 1495–1517, Sep. 2002.

[5] S. Tsao and C. H. Huang, "A survey of energy efficient MAC protocols for IEEE 802.11 WLAN," *Comput. Commun.*, vol. 34, no. 1, pp. 54–67, Jan. 2011.

[6] I. Humar, X. Ge, L. Xiang, M. Jo, and M. Chen, "Rethinking energy efficiency models of cellular networks with embodied energy," *IEEE Netw. Mag.*, vol. 25, no. 3, pp. 40–49, Mar./Apr. 2011.

[7] L. Xiang, X. Ge, C.-X. Wang, F. Li, and F. Reichert, "Energy efficiency evaluation of cellular networks based on spatial distributions of traffic load and power consumption," *IEEE Trans. Wireless Commun.*, vol. 12, no. 3, pp. 961–973, Mar. 2013.

[8] X. Ge, K. Huang, C.-X. Wang, X. Hong, and X. Yang, "Capacity analysis of a multi-cell multi-antenna cooperative cellular network with co-channel interference," *IEEE Trans. Wireless Commun.*, vol. 10, no. 10, pp. 3298–3309, Oct. 2011.

[9] L. Liu, X. Cao, Y. Cheng, L. Du, W. Song, and Y. Wang, "Energy-efficient capacity optimization in wireless networks," in *Proc. IEEE INFOCOM*, 2014 . [Online]. Available: http://www.ieee-infocom.org/Program_technical.html, to be published.

[10] *IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-2007, Jun. 2007.

[11] *IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements*, IEEE Std 802.11e-2005, 2005.

[12] H. Li, Y. Cheng, C. Zhou, and W. Zhuang, "Routing metrics for minimizing end-to-end delay in multi-radio multi-channel wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 11, pp. 2293–2303, Nov. 2013.

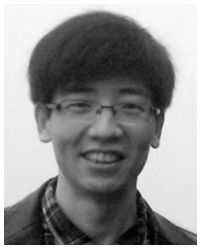
[13] V. Namboodiri and L. Gao, "Energy-efficient VoIP over wireless LANs," *IEEE Trans. Mobile Comput.*, vol. 9, no. 4, pp. 566–581, Apr. 2010.

[14] Y. He, R. Yuan, and W. Gong, "Modeling power saving protocols for multicast services in 802.11 wireless LANs," *IEEE Trans. Mobile Comput.*, vol. 9, no. 5, pp. 657–671, May 2010.

[15] Y. He and R. Yuan, "A novel scheduled power saving mechanism for 802.11 wireless LANs," *IEEE Trans. Mobile Comput.*, vol. 8, no. 10, pp. 1368–1383, Oct. 2009.
[16] Z. Zeng, Y. Gao, and P. R. Kumar, "Sofa: A sleep-optimal fair-attention scheduler for the power-saving mode of WLANs," in *Proc. IEEE ICDCS*, 2011, pp. 87–98.
[17] X. Prez-Costa and D. Camps-Mur, "IEEE 802.11E QoS and power saving features overview and analysis of combined performance," *IEEE Wireless Commun.*, vol. 17, no. 4, pp. 88–96, Aug. 2010.
[18] T. Lee and J. Hsieh, "Low complexity class-based scheduling algorithm for scheduled automatic power-save delivery for wireless LANs," *IEEE Trans. Mobile Comput.*, vol. 12, no. 3, pp. 571–580, Mar. 2013.
[19] X. Prez-Costa and D. Camps-Mur, "AU-APSD: Adaptive IEEE 802.11e unscheduled automatic power save delivery," in *Proc. IEEE ICC*, 2006, pp. 2020–2027.
[20] X. Chen, S. Jin, and D. Qiao, "M-PSM: Mobility-aware power save mode for IEEE 802.11 WLANs," in *Proc. IEEE ICDCS*, 2011, pp. 77–86.
[21] S. Mangold, S. Choi, G. R. Hiertz, O. Klein, and B. Walke, "Analysis of IEEE 802.11e for QoS support in wireless LANs," *IEEE Wireless Commun.*, vol. 10, no. 6, pp. 40–50, Dec. 2003.
[22] X. Lin, L. Liu, J. Liu, N. Xie, and H. Wang, "Locating the optimal sleep window for enhancing the energy efficiency of VoIP in WiMAX systems: A modified analysis model and performance evaluation," in *Proc. 7th Int. Conf. WiCOM*, 2011, pp. 1–5.
[23] X. Lin, L. Liu, H. Wang, and Y. Kwok, "On exploiting the on–off characteristics of human speech to conserve energy for the downlink VoIP in WiMAX systems," in *Proc. 7th IWCMC*, 2011, pp. 337–342.
[24] J. Lee and D. Cho, "Dual power-saving modes for voice over IP traffic supporting voice activity detection," *IET Commun.*, vol. 3, no. 7, pp. 1239–1249, Jul. 2009.
[25] A. Pyles, Z. Ren, G. Zhou, and X. Liu, "SiFi: Exploiting VoIP silence for WiFi energy savings in smart phones," in *Proc. 13th Int. Conf. Ubiquitous Comput.*, 2011, pp. 325–334.
[26] L. Wang and H. Xu, "An energy saving based uplink scheduling algorithm for VoIP services in IEEE 802.16 systems," in *Proc. 4th Int. Conf. Wireless Commun., Netw. Mobile Comput.*, 2008, pp. 1–4.
[27] K. Ting, F. Kuo, B. Hwang, H. Wang, and C. Tseng, "A power-saving and robust point coordination function for the transmission of VoIP over 802.11," in *Proc. ISPA*, 2010, pp. 283–289.
[28] G. Wong, Q. Zhang, and D. Tsang, "Switching cost minimization in the IEEE 802.16 e mobile WiMAX sleep mode operation," *Wireless Commun. Mobile Comput.*, vol. 10, no. 12, pp. 1576–1588, 2010.
[29] The Network Simulator - ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/

**Lu Liu** (S'13) received the B.S. degree in automation in 2010 from Tsinghua University, Beijing, China, and the M.S. degree in electrical engineering in 2012 from Illinois Institute of Technology, Chicago, IL, USA, where she is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering.

Her current research interests include energy-efficient networking and communication, resource allocation, and protocol design of wireless networks.



**Xianghui Cao** (S'08–M'11) received the B.S. and Ph.D. degrees in control science and engineering from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively.

During 2007–2009, he was a Visiting Scholar with the Department of Computer Science, The University of Alabama, Tuscaloosa, AL, USA. He is currently with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. His research interests include wireless network performance analysis, energy efficiency of wireless networks, networked estimation and control, and network security.

Dr. Cao is or was a Technical Program Committee Member of the IEEE Global Communications Conference (GLOBECOM) in 2013 and 2014, the IEEE International Conference on Communications in 2014, the IEEE Vehicular Technology Conference in 2013 and 2014, as well as other conferences. He is also an Associate Editor of the *KSII Transactions on Internet and Information Systems and Security and Communication Networks* (Wiley).

**Yu Cheng** (S'01–M'04–SM'09) received the B.E. and M.E. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2003.

From September 2004 to July 2006, he was a Postdoctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON. Since August 2006, he has been with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA, where he is currently an Associate Professor. His research interests include next-generation Internet architectures and management, wireless network performance analysis, network security, and wireless/wireline interworking.

Dr. Cheng served as a Co-Chair of the Wireless Networking Symposium of the IEEE International Conference on Communications (ICC) in 2009, a Co-Chair of the Communications QoS, Reliability, and Modeling Symposium of the IEEE Global Communications Conference (GLOBECOM) in 2011, a Co-Chair of the Signal Processing for Communications Symposium of the IEEE ICC in 2012, a Co-Chair of the Ad Hoc and Sensor Networking Symposium of the IEEE GLOBECOM in 2013, and a Technical Program Committee Co-Chair of the International Conference on Wireless Algorithms, Systems, and Applications (WASA) in 2011. He is a founding Vice Chair of the IEEE Communications Society Technical Subcommittee on Green Communications and Computing. He is an Associate Editor of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY and the New Books and Multimedia Column Editor of *IEEE Network*. He received a Best Paper Award at QShine in 2007 and at IEEE ICC in 2011. He received the National Science Foundation CAREER Award in 2011 and the IIT Sigma Xi Research Award in the junior faculty division in 2013.

**Zhisheng Niu** (M'98–SM'99–F'12) received the B.E. degree from Beijing Jiaotong University, Beijing, China, in 1985 and the M.E. and D.E. degrees from Toyohashi University of Technology, Toyohashi, Japan, in 1989 and 1992, respectively.

During 1992–1994, he was with Fujitsu Laboratories, Ltd., Kawasaki, Japan. In 1994, he joined Tsinghua University, Beijing, China, where he is currently a Professor with the Department of Electronic Engineering, the Deputy Dean of the School of Information Science and Technology, and the Director of Tsinghua-Hitachi Joint Laboratory on Environmental Harmonious ICT. He is also a guest Chair Professor with Shandong University, Jinan, China. His major research interests include queuing theory, traffic engineering, mobile Internet, radio resource management of wireless networks, and green communication and networks.

Dr. Niu is a Fellow of the Institute of Electronics, Information, and Communication Engineers. He received the Outstanding Young Researcher Award from the National Natural Science Foundation of China in 2009 and the Best Paper Award from the IEEE Communication Society Asia-Pacific Board in 2013. He was also a co-recipient of the Best Paper Awards at the Asia-Pacific Conference on Communication in 2007, 2009, and 2013, as well as from the International Conference on Wireless Communications and Signal Processing in 2013 and the Best Student Paper Award from the 25th International Teletraffic Congress (ITC25). He is currently the Chief Scientist of the National Basic Research Program (the "973 Project") of China on "Fundamental Research on the Energy and Resource Optimized Hyper-Cellular Mobile Communication System" (2012–2016), which is the first national project on green communications in China.