Computer Networks 55 (2011) 3721–3733

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Per-flow sleep scheduling for power management in IEEE 802.16 wireless networks

Jen-Jee Chen [a], Shih-Lin Wu [b,*], Shiou-Wen Wang [b], Yu-Chee Tseng [c,d]

[a] Department of Electrical Engineering, National University of Tainan, Tainan 70005, Taiwan
[b] Department of Computer Science and Information Engineering, Chang Gung University, Kweishan Taoyuan 33302, Taiwan
[c] Department of Computer Science, National Chiao Tung University, Hsin-Chu 30010, Taiwan
[d] Research Center for Information Technology Innovation, Academia Sinica, Taipei 11529, Taiwan

ARTICLE INFO

Article history:
Available online 30 March 2011

Keywords:
IEEE 802.16
Mobile communication
Power saving class
Quality of service (QoS)
WiMAX
Wireless network

ABSTRACT

Power management is a critical issue in IEEE 802.16 wireless networks. In the standard, a *power saving class* (*PSC*) of type II is defined to support real-time traffic flows. It allows a flow to switch periodically between active and sleep states to save energy. However, previous studies either consider adjusting start frames of PSCs by assuming that the PSCs are already given or assume one single PSC to accommodate all flows in a mobile station, thus leading to higher energy cost. This paper proposes two "per-flow" sleep scheduling schemes, which assign one PSC to each real-time flow according to its QoS parameters. This leads to less energy consumption, more efficient use of bandwidth, and more compact listening windows. We also prove that deciding whether a given scheduling problem is solvable can be reduced to a maximum matching problem, which is computationally tractable. Simulation results show that such a per-flow scheduling does perform much closer to the active ratio lower bound and achieve higher resource utilization than previous schemes.

1. Introduction

The IEEE 802.16 [1] has been defined for broadband wireless access for *mobile stations* (*MSs*). One important design issue in IEEE 802.16 is *power saving classes* (*PSCs*), which allow an MS to switch between active and sleep modes to reduce unnecessary energy consumption. The standard defines three types of PSCs for different traffic characteristics. For type I, the sizes of listening windows are fixed while the sizes of sleep windows grow exponentially if no data packet arrives. Once a data packet arrives, the corresponding PSC is deactivated. This type is suitable for non-real-time variable-rate (NRT-VR) and Best-Effort (BE) connections. For type II, both listening and sleep windows are of fixed sizes. Such a PSC is deactivated only when asked. This type is suitable for Unsolicited Grant Service (UGS) and real-time variable-rate (RT-VR) connections. Type III is only valid for one sleep window, after which the PSC is deactivated. It is suitable for multicast and management operations. The standard suggests that connections with similar characteristics can be associated with one PSC. If there are multiple PSCs between an MS and a BS, the MS can go to sleep only if all its PSCs are in sleep windows. Fig. 1 shows an example with three PSCs in an MS and the actual intervals that the MS can go to sleep. Clearly, we observe that by increasing the overlapping of listening windows, the MS's energy consumption can be reduced. However, this is left as an open issue for designers.

Intensive works have been devoted to the power saving issues of IEEE 802.11 networks [2–4]. However, these techniques can not be directly applied to IEEE 802.16 because IEEE 802.11 is a CSMA/CA-based (Carrier Sense Multiple Access with Collision Avoidance-based) wireless access protocol. Analyses of IEEE 802.16 networks' energy costs are in [5–8]. These results have provided a potential guidance for setting PSCs' parameters. However, these schemes

* Corresponding author.
E-mail addresses: jjchen@mail.nutn.edu.tw (J.-J. Chen), slwu@mail.cgu.edu.tw (S.-L. Wu), m9529004@stmail.cgu.edu.tw (S.-W. Wang), yctseng@cs.nctu.edu.tw (Y.-C. Tseng).

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.03.003

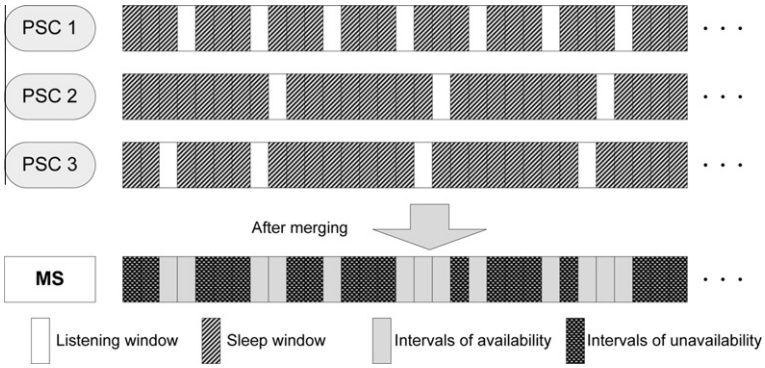

Fig. 1. Sleep frame determination of an MS.

all assume non-real-time traffic and most of them consider the arrival patterns to be memoryless, which is not always true in the real world. Numerous works in the literature [9–13] focus on the designs of PSCs of type I. How to adaptively adjust the initial sleep window is addressed in [9]. The relationship between the initial sleep window and the estimated packet inter-arrival time is studied in [10]. How to adjust the initial and the maximum sleep windows is studied in [11]. Assuming that the distribution of the response packet arrival time is known, [12] proposes a decision algorithm. How to adjust the minimum and the maximum sleep windows is discussed in [13]. For PSCs of type II, a Maximum Unavailability Interval (MUI) scheme is proposed in [14] for selecting the optimal start frames of PSCs to maximize the unavailable time. This work does not answer how to decide PSCs' parameters. In [15,16], since one single PSC is applied to serve all real-time connections in an MS, the selection of sleep and listening windows must meet the strictest bandwidth and packet delay bound requirements of all connections. This could incur waste of bandwidth and extra listening windows.

Given a set of real-time flows between an MS and a BS, this paper considers the "per-flow" sleep scheduling problem. This involves not only the selection of PSC parameters of all flows to meet their QoS, but also the scheduling of their active times to reduce the overall duty cycle. This may outperform the results in [15,16] since using one PSC for each flow can more accurately capture its traffic characteristic (Section 2 will illustrate this by an example.) Our results are thus more bandwidth- and energy-efficient. Two standard-compliant schemes are proposed. Moreover, we prove that deciding whether a given scheduling problem is solvable can be reduced to a maximum matching problem in a bipartite graph which can be solved in polynomial time. Finally, simulation results are provided to verify these claims.

The rest of this paper is organized as follows. Section 2 shows our motivation and the problem definition. Section 3 presents our schemes. In Section 4, we conduct some feasibility study of the scheduling problem. Simulation results are in Section 5. Section 6 concludes this paper.

2. Motivation and problem definition

We first motivate our work and then present our problem definition. For PSCs of type II, previous works [15,16] try to form one single PSC to serve all connections in an MS. While this is simpler, there are several drawbacks. First, the strictest delay bound among all connections must be followed to avoid missing any deadlines. Second, in each listening window, the BS needs to reserve the maximum bandwidth to accommodate the maximum possible load. This leads to waste of bandwidth and extra awake frames. If each connection has its own PSC according to its packet inter-arrival time and delay bound, the MS only needs to stay awake at proper times, thus incurring less resource waste. Fig. 2 shows an example with two connections C_1 and C_2, which have packet inter-arrival times of $PI_1 = 3F$ and $PI_2 = 6F$, delay bounds of $D_1 = 6F$ and $D_2 = 18F$, and expected packet sizes of $S_1 = \frac{2}{5}B$ and $S_2 = \frac{2}{5}B$, respectively, where F is the frame duration and B is the maximum available resource per frame for the MS. As Fig. 2(a) shows, if the *periodic on–off scheme (PS)* [15] is applied, there is only one PSC of type II, which has a sleeping cycle of 6 frames (i.e., the strictest delay bound $6F$ by C_1) and a listening window of 2 frames per cycle (i.e., the traffic demand of C_1 and C_2 per $6F$, $\lceil \frac{6F}{3F} \rceil S_1 + \lceil \frac{6F}{6F} \rceil S_2 = \frac{6}{5}B$, after taking the ceiling function). As Fig. 2(b) shows, by using two PSCs, we can construct a PSC P_1 of period $T_1 = 6$ frames for C_1 and a PSC P_2 of period $T_2 = 18$ frames for C_2. P_1 and P_2 need 1 and 2 frames of listening windows per cycle, respectively (note that two PSCs can share some active frames as long as their traffics can be consumed). Compared to Fig. 2(a), which needs 2 listening frames per 6 frames, Fig. 2(b) needs 4 listening frames per 18 frames, thus reducing the energy cost by 33.3%.

This paper considers the per-flow sleep scheduling problem as follows:

Given

(1) n real-time connections C_i, $i = 1..n$, between an MS and a BS, where each connection C_i has a packet inter-arrival time PI_i (ms), a delay bound for packets D_i (ms), and an expected packet size S_i (bits) and
(2) the BS can allocate to the MS at most B (bits) per frame

Find

n PSCs P_i, $i = 1..n$, each is assigned to the connection C_i and with sleeping cycle T_i (frames), listening window T_i^L (frames), and starting frame of the first listening window T_i^S.

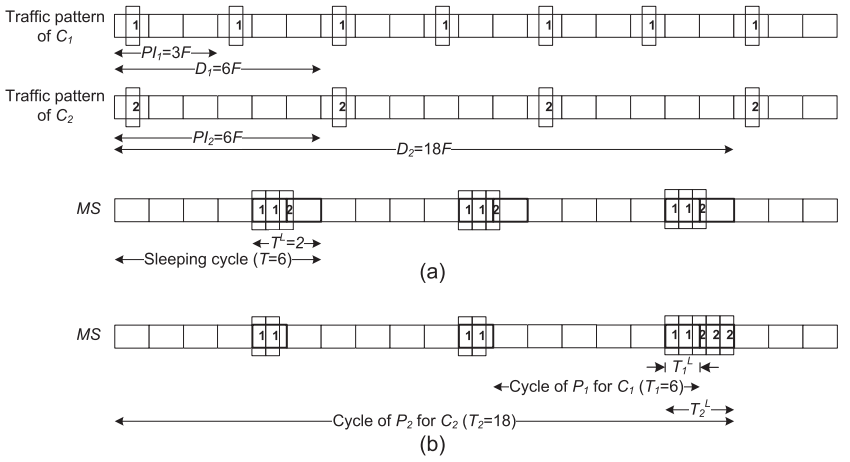

Fig. 2. Sleep scheduling for two connections C_1 and C_2 using (a) one PSC and (b) two PSCs.

Minimize the ratio of active frames for the MS
Subject to

(1) $T_i \times F \leqslant D_i$ (delay bound) and
(2) $\lceil \lceil \frac{T_i \times F}{PI_i} \rceil \times S_i / B \rceil \leqslant T_i^L$ (bandwidth requirement), $i = 1..n$

3. Proposed schemes

Next, we present our *per-flow sleep scheduling* (*PSS*) algorithm. Our goal is to determine the following parameters for each C_i, $i = 1..n$: (1) the cycle length T_i, (2) the listening window T_i^L in each cycle, (3) the starting frame T_i^S of the first listening window, and (4) the amount of resource $R_{i,j}$ to be allocated to the jth active frame of each cycle, $j = 1..T_i^L$. The calculation should be done at the BS side. Our scheme will maintain a property that the cycle length T_i of each C_i is an integer multiple of the previous T_{i-1}. Therefore, we will sometimes call T_1 the basic cycle T_{basic}. The goal is to increase the overlapping of these listening windows so as to reduce the MS's duty cycle. Assuming that T_{basic} is known, our PSS algorithm involves an iterative process for $i = 1..n$, where each iteration has two steps: (i) determine T_i of each C_i and (ii) schedule $R_{i,j}$, T_i^L, and T_i^S of each C_i. We will discuss how to determine the basic cycle T_{basic} later on. Below, we present some observations, which will serve as guidelines for our design.

Observation 1. *The resource of bandwidth B bits per frame allocated to an MS can be regarded as an infinite sequence S of a period of one frame. S can be divided into p sub-sequences S_i^p, each with a period of p frames and a resource of B bits per cycle, where i = 1..p and p is a positive integer. Alternatively, S can be divided into m sub-sequences S_i^{p,k_i}, each with a period of p and a resource of $k_i \times B$ bits per cycle, where i = 1..m, m < p, k_i is a positive integer, and $\sum_{i=1..m} k_i = p$.*

Observation 2. *A sub-sequence S_i^p can be further divided into p' sub-sequences $S_{i,j}^{p \times p'}$, each with a period of $p \times p'$ frames and each shifted by a distance of p, where j = 1..p′. Similarly, a sub-sequence S_i^{p,k_i} can be further divided into p' sub-sequences $S_{i,j}^{p \times p', k_i}$, j = 1..p′, with similar properties except that in each cycle the amount of resource is $k_i \times B$ bits.*

Observation 3. *The scheduling problem for a connection C_i can be regarded as placing its demand on a sub-sequence with a proper period and resource per cycle. We propose two strategies toward this goal. The first one, called delay bound-based (DB-based) strategy, tries to accumulate a connection's traffic as much as possible until reaching the delay bound and serve the connection by a sub-sequence with a period slightly tighter than the delay bound and a resource sufficient for the accumulated traffics per cycle. In this way, there are less active frames incurred by the connection. The second one, called packet inter-arrival-based (PI-based) strategy, tries to serve a connection's traffic immediately once a packet arrives by a sub-sequence with a period slightly tighter than the packet inter-arrival time. If each packet size is small, we may overlap multiple connections' active frames and serve their traffic by one or few active frames.*

Observations 1 and 2 indicate some ways to decompose the resource allocated to an MS. Fig. 3(a) shows two examples. With $p = 3$, S is divided into three subsequences S_1^3, S_2^3, and S_3^3, each with a period of 3 frames. Alternatively, with $m = 2$, S can be divided into two subsequences $S_1^{3,2}$ and $S_2^{3,1}$, which have $2B$ and B bits per cycle, respectively. Following Observation 2, Fig. 3(b) shows how to decompose S_1^3 into $p' = 2$ sub-sequences $S_{1,1}^{3\times2}$ and $S_{1,2}^{3\times2}$, each with a period of 3×2 frames, and how to decompose $S_1^{3,2}$ into $S_{1,1}^{3\times2,2}$, and $S_{1,2}^{3\times2,2}$.

Our PSS follows these observations to pack the traffic of connections together to reduce energy consumption. It has two steps and adds C_i one-by-one to the MS such that the additional active frames for the MS are as few as possible. Analyzing this for each C_i, we will determine the following four sleeping parameters: the sleeping cycle T_i, the starting frame of the first listening window T_i^S, the listening window T_i^L, and the amount of resource $R_{i,j}$ to be allocated to the jth active frame of each sleeping cycle, $j = 1..T_i^L$. A data structure π_k, $k = 1..\frac{T_n}{T_{basic}}$, is maintained to record the amount of remaining free resource in the kth basic cycle.

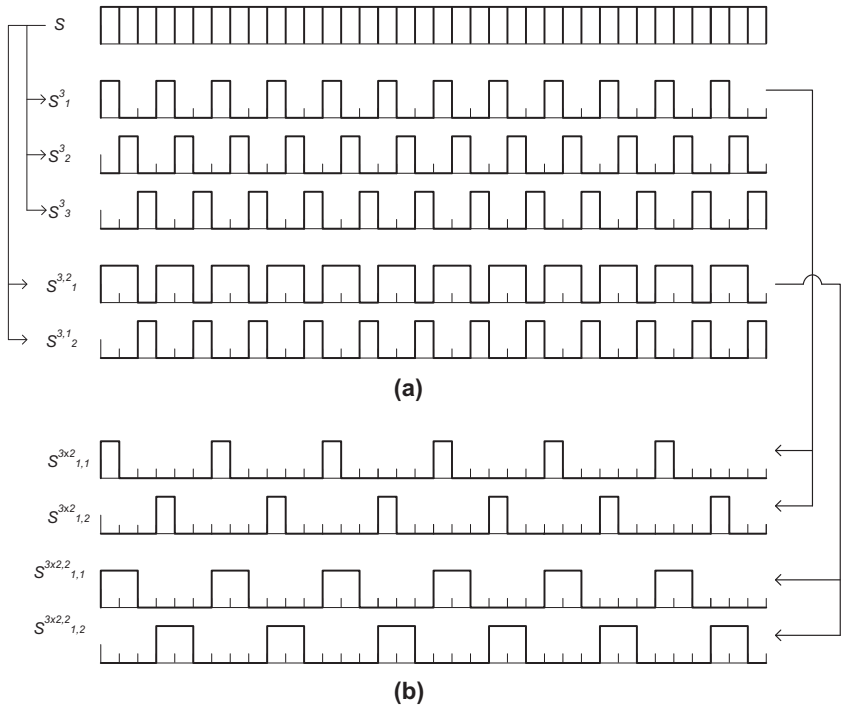

Fig. 3. Examples of (a) Observation 1 and (b) Observation 2.

Initially, $\pi_k = B \times T_{basic}$. Fig. 4 shows an example. In the example, a connection C_i is scheduled and assigned $T_i = 6$ (frames), $T_i^L = 2$ (frames), $T_i^S = 4$, $R_{i,1} = 0.6B$, and $R_{i,2} = B$, where $T_{basic} = 3$ frames. On the other hand, the BS records the remaining free resources of the 2nd, 4th, and 6th basic cycles as $\pi_2 = \pi_4 = \pi_6 = B$ and the 1st, 3rd, and 5th basic cycles as $\pi_1 = \pi_3 = \pi_5 = 2.6B$. Note that there is $0.4B$ bandwidth in each basic cycle being consumed by other connections scheduled before C_i.

Step (1) Determining T_i of each C_i: We propose two approaches for this step. The first one, called *PSS-DB* (*PSS by delay bound*), sorts C_is by their packet delay bounds such that $D_1 \leqslant D_2 \leqslant \cdots \leqslant D_n$. The second one, called *PSS-PI* (*PSS by packet inter-arrival time*), sorts C_is by their packet inter-arrival times such that $PI_1 \leqslant PI_2 \leqslant \cdots \leqslant PI_n$. The design philosophy is in accordance with Observation 3.

For PSS-DB, we let $T_1 = T_{basic}$ and set T_i for $i = 2..n$ as follows:

$$T_i = T_{i-1} \times \left\lfloor \frac{D_i}{T_{i-1} \times F} \right\rfloor. \quad (1)$$

Eq. (1) sets T_i as a positive integer, a multiple of the previous T_{i-1}. In fact, our assignment guarantees in a recursive manner that $T_i \leqslant \lfloor \frac{D_i}{F} \rfloor$. The initial T_1 would satisfy this condition (to be shown later on). Since $D_{i-1} \leqslant D_i$, $\lfloor \frac{D_i}{T_{i-1} \times F} \rfloor$ in Eq. (1) must be a positive integer. Also, Eq. (1) implies that $T_i \leqslant T_{i-1} \times \frac{D_i}{T_{i-1} \times F} = \frac{D_i}{F}$. Since T_i is an integer, $T_i \leqslant \lfloor \frac{D_i}{F} \rfloor$ meets the delay bound for C_i.

For PSS-PI, we assume that $PI_i \leqslant D_i$ (this is usually true for most of delay-tolerant real-time applications). We let $T_1 = T_{basic}$ and set T_i for $i = 2..n$ as follows:

$$T_i = \max\left\{ T_{basic}, T_{i-1} \times \left\lfloor \frac{PI_i}{T_{i-1} \times F} \right\rfloor \right\}. \quad (2)$$

Eq. (2) also sets T_i as a positive integer multiple of the previous T_{i-1}. Our assignment guarantees in a recursive

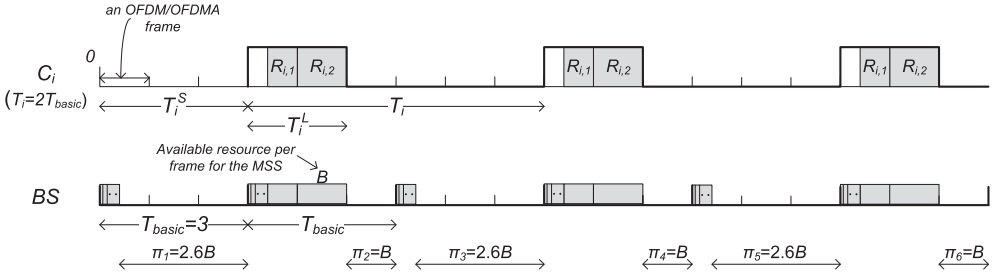

Fig. 4. Output parameters T_i, T_i^L, T_i^S, and $R_{i,j}$ after scheduling C_i and the system parameters π_k, $k = 1..\frac{T_n}{T_{basic}}$, maintained by the central BS.

manner that $T_i = T_{basic} \leqslant \left\lfloor \frac{\min\{D_i\}}{F} \right\rfloor$ if $PI_i \leqslant \min_{i=1..n}\{D_i\}$ and $T_i \leqslant \left\lfloor \frac{PI_i}{F} \right\rfloor$ otherwise. The initial T_1 would satisfy the former (to be shown later). Since PI_is are sorted in an ascending order, Eq. (2) will force those C_is such that $PI_i \leqslant \min_{i=1..n}\{D_i\}$ to choose their $T_i = T_{basic}$. The rest of the C_is will satisfy the later condition since $T_i \leqslant T_{i-1} \times \frac{PI_i}{T_{i-1} \times F} = \frac{PI_i}{F}$. It follows that all C_is will meet their delay bounds because $PI_i \leqslant D_i$.

Theorem 1. *In both PSS-DB and PSS-PI, it is guaranteed that each T_i is a positive integer multiple of $T_1 = T_{basic}$, $i = 2..n$, and each C_i's cycle meets its delay bound, i.e., $T_i \leqslant \left\lfloor \frac{D_i}{F} \right\rfloor$, $i = 1..n$.*

Step (2) Scheduling $R_{i,j}$, T_i^L, and T_i^S of each C_i: This step is the same for PSS-DB and PSS-PI. So we will not distinguish between them. Recall the data structure π_k, $k = 1..\frac{T_n}{T_{basic}}$. We will sequentially schedule C_i, $i = 1..n$, by updating π_k. Specifically, when C_i is under consideration, we will pick one basic cycle among all $\frac{T_n}{T_{basic}}$ basic cycles as the starting point and examine the subsequent basic cycles. For each basic cycle being examined, its remaining resource is allocated; this is repeated until we have allocated sufficient resource for C_i. Among all starting points, the one which causes the least increment on the number of active frames is selected. The detail procedure for placing C_i's demand is as follows:

(a) Calculate the required resource γ_i of C_i per T_i by

$$\gamma_i = \left\lceil \frac{T_i \times F}{PI_i} \right\rceil \times S_i. \quad (3)$$

(b) Recall that each T_j is an integer multiple of T_{j-1}, $j = 2..n$. So after placing C_1's, C_2's, …, C_{i-1}'s demands, the sequence π_k, $k = 1..\frac{T_n}{T_{basic}}$, has a period of $\frac{T_{i-1}}{T_{basic}}$. To place C_i's demand, we only need to check the first $\frac{T_i}{T_{basic}}$ basic cycles as T_i's starting point. Specifically, for $k = 1$ to $\frac{T_i}{T_{basic}}$ with $\pi_k > 0$, we compute a cost function $f(k)$ to represent the extra active frames incurred to the MS if we place C_i's demand on the kth and subsequent basic cycles. Note that since the listening window of a PSC must be continuous, the resources allocated to C_i must be continuous (i.e., we will not leave a frame unallocated if there are frames being allocated before and after the frame).

(c) Let k^* be the index which induces the smallest cost function $f(k)$ in step b. We will place C_i's demand starting from the k^*th basic cycle. In case that there is a tie, we will give priority to the one which leaves the least remaining resource in the last frame where C_i's demand is placed.

(d) Then we set T_i^S to the index of the first frame where C_i's demand is placed and set T_i^L to the number of frames from the first to the last frame where C_i's demand is placed. Also, $R_{i,j}$ is set accordingly. Finally, we update the remaining free resources in π_k, $k = 1..n$, by subtracting from them the amounts of resources allocated to C_i (note that since the period is T_i, $\pi_k = \pi_{k+\ell\frac{T_i}{T_{basic}}}$, $\ell = 1..\left(\frac{T_n}{T_i} - 1\right)$).

Example 1. Fig.5 shows an example of step 2.The MS contains 4 connections C_1, C_2, C_3, and C_4 with sleeping cycles of $T_1 = T_{basic}$, $T_2 = 2T_{basic}$, $T_3 = 2T_{basic}$, and $T_4 = 4T_{basic}$ and

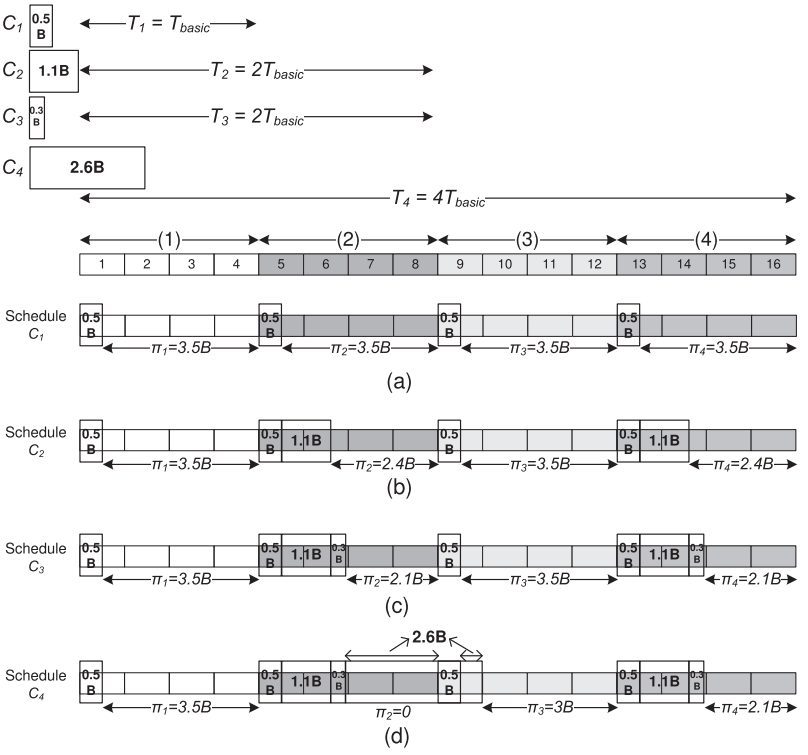

Fig. 5. Example of scheduling $R_{i,j}$, T_i^L, and T_i^S of four connections in the MS.

required resources per cycle of $\gamma_1 = 0.5B$, $\gamma_2 = 1.1B$, $\gamma_3 = 0.3B$, and $\gamma_4 = 2.6B$, respectively, where $T_{basic} = 4$ frames.Initially, $\pi_k = 4B$ for $k = 1..4$. Then, each C_i is scheduled as follows.For C_1, any selection of k^* is the same for it.So we set $k^* = 1$, $R_{1,1} = 0.5B$, $T_1^S = 1$, and $T_1^L = 1$.The BS reserves $\gamma_1 = 0.5B$ resource for C_1 in every basic cycle as shown Fig.5(a), so $\pi_1 = \pi_2 = \pi_3 = \pi_4 = 3.5B$. For C_2, its k^* can be 1 or 2.Allocating γ_2 in the first or second basic cycle would add one more active frame to the MS (i.e., $f(1) = f(2) = 1$) and leave the same remaining resource of $0.4B$ in the last frame.So we randomly select $k^* = 2$, which gives $R_{2,1} = 0.5B$, $R_{2,2} = 0.6B$, $T_2^S = 5$, and $T_2^L = 2$, as shown in Fig.5(b).Then we update $\pi_1 = 3.5B$, $\pi_2 = 2.4B$, $\pi_3 = 3.5B$, and $\pi_4 = 2.4B$. For C_3, choosing $k^* = 1$ or 2 would require no additional active frame for the MS (i.e., $f(1) = f(2) = 0$).But setting $k^* = 2$ would leave less remaining resource in the last frame (i.e., $0.1B$). So we set$k^* = 2$ and update $R_{3,1} = 0.3B$, $T_3^S = 6$, and $T_3^L = 1$, as shown in Fig. 5(c). Then we update $\pi_1 = 3.5B$, $\pi_2 = 2.1B$, $\pi_3 = 3.5B$, and $\pi_4 = 2.1B$. For C_4, setting $k^* = 2$ or 4 would add less active frames (i.e., $f(2) = f(4) = 2 < f(1) = f(3) = 3$). Since $k^* = 2$ and 4 will both leave the same remaining resource in the last frame, we randomly pick $k^* = 2$. So we set $R_{4,1} = 0.1B$, $R_{4,2} = 1B$, $R_{4,3} = 1B$, $R_{4,4} = 0.5B$, $T_4^S = 6$, and $T_4^L = 4$, as shown in Fig. 5(d). Then we update $\pi_1 = 3.5B$, $\pi_2 = 0$, $\pi_3 = 3B$, and $\pi_4 = 2.1B$.

Example 2. Fig. 6 uses an example to compare PSS-DB and PSS-PI. In Fig. 6(a), the MS contains 3 connections C_1, C_2, and C_3 with packet inter-arrival times of $PI_1 = 10$ ms, $PI_2 = 30$ ms, and $PI_3 = 30$ ms, delay bounds of $D_1 = 20$ ms, $D_2 = 100$ ms, and $D_3 = 100$ ms, and packet sizes of $S_1 = \frac{B}{2}$, $S_2 = \frac{B}{2}$, and $S_3 = \frac{B}{2}$. Fig. 6(b) changes the packet sizes to $S_1 = \frac{B}{4}$, $S_2 = \frac{B}{4}$, and $S_3 = \frac{B}{4}$. Fig. 6(a) shows that PSS-DB will consume 9 active frames per 20 frames and PSS-PI will consume 10 active frames per 20 frames, while Fig. 6(b) shows that PSS-DB will consume 7 active frames per 20 frames and PSS-PI will consume only 5 active frames per 20 frames. Intuitively, PSS-DB can pack packets of a connection together according to their delay bound and is more favorable when S_is are relatively closer to B. Contrarily, PSS-PI tries to pack packets of different connections together and serve them immediately after their arrivals and is more favorable when S_is are relatively smaller than B.

Lastly, we present how to determine the basic cycle T_{basic}. Since $T_{basic} = T_1$ and T_1 must satisfy $T_1 \leqslant \lfloor \frac{D_1}{F} \rfloor$ for PSS-DB and $T_1 \leqslant \left\lfloor \frac{\min_{i=1..n} D_i}{F} \right\rfloor$ for PSS-PI, we propose to pick T_{basic} from the interval $\left[1, \lfloor \frac{D_1}{F} \rfloor\right]$ for PSS-DB and the interval $\left[1, \left\lfloor \frac{\min_{i=1..n} D_i}{F} \right\rfloor\right]$ for PSS-PI. For each candidate T_{basic}, we adopt an exhausted search to compute a cost function $g(T_{basic})$ to represent the ratio of active frames of the MS if T_{basic} is used. Let T^*_{basic} be the basic cycle which induces the least cost. Then this T^*_{basic} is chosen for T_1.

4. Feasibility study of the scheduling problem

The above discussion did not answer the question: "What happens when a feasible scheduling can not be found?" Below, we conduct a feasibility study of the sleep scheduling problem. It is not hard to see that given a set of connections, if their traffic can be satisfactorily arranged without violating their deadlines, then "always active" is a straightforward schedule (note that this statement does not imply whether the scheduling is energy-efficient or not). Below, we show that deciding whether traffic of a set of connections can be satisfactorily scheduled can be reduced to a maximum matching problem, which is computationally tractable. Solutions for maximum matching

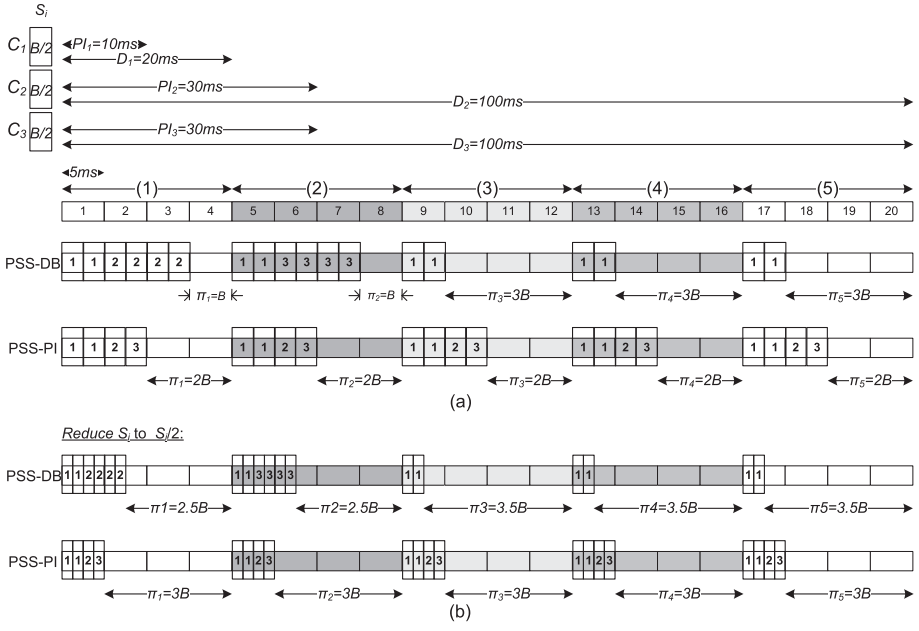

Fig. 6. Comparisons of PSS-DB and PSS-PI under different packet sizes: (a) $S_i = \frac{B}{2}$ and (b) $S_i = \frac{B}{4}$.

can be found in [17]. Thus, deciding whether a scheduling problem is feasible has a polynomial-time solution (following the above note, it remains a question whether the solution is most energy-efficient).

We are still given C_i and its PI_i, D_i, and S_i, $i = 1..n$. In our derivation, we assume that the units of PI_i and D_i are frames and the units of S_i and the resource B are bits. Let $L = lcm\{PI_i, i = 1..n\}$. Note that traffic arrivals repeat at the period of L frames. Below, we will model our scheduling problem as a maximum matching problem in a bipartite graph $G = (\{V_l, V_r\}, E)$ as defined below.

1. For each C_i, consider its packet arrivals during L frames. There are $\frac{L}{PI_i} \times S_i$ bits. So we construct for C_i the same number of vertices, denoted by $C_{i,j,k}$, $j = 1..\frac{L}{PI_i}$ and $k = 1..S_i$, in V_l. So $|V_l| = \sum_{i=1}^{n} \left(S_i \times \frac{L}{PI_i}\right)$.
2. For the B bits in continuous L frames, we construct the same number of vertices, denoted by $F_{x,y}$, $x = 1..L$ and $y = 1..B$, in V_r. So $|V_r| = B \times L$.
3. We regard vertex $C_{i,j,k} \in V_l$ as the kth bit of the jth packet of connection C_i. Let $t_{i,j}$ be the frame index of its arrival. Then it has to be delivered by frame $t_{i,j} + D_i - 1$. So we construct edges $(C_{i,j,k}, F_{x,y})$ in E for $x = t_{i,j} + 1..t_{i,j} + D_i - 1 (mod\ L)$ and $y = 1..B$. Each edge means that $C_{i,j,k}$ can be assigned to resource $F_{x,y}$ without violating the delay bound. Note that some x may exceed L, so we use "*mod L*" to represent the subsequent L frames in the next round, i.e., they are represented by wrap-around edges.

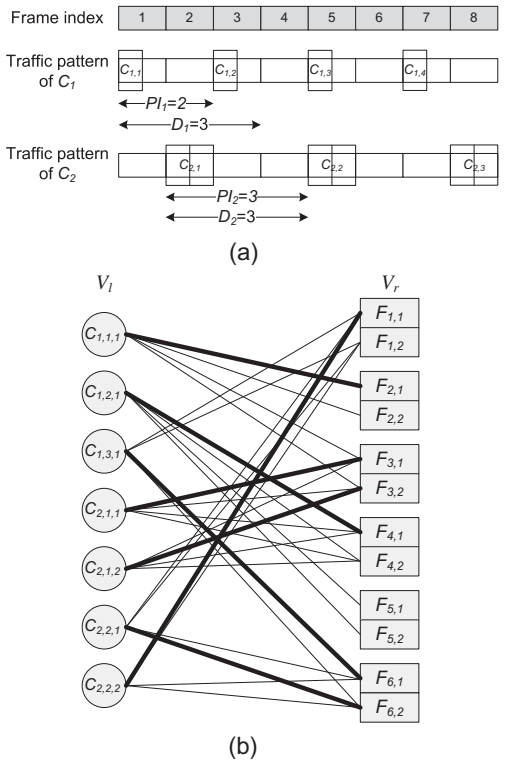

Fig. 7. Example of modeling a scheduling problem as a maximum matching problem.

Example 3. Fig. 7(a) shows an example. The MS contains 2 connections C_1 and C_2 with $PI_1 = 2$, $PI_2 = 3$, $D_1 = 3$, $D_2 = 3$, $S_1 = 1$, and $S_2 = 2$. Assuming $B = 2$, we can form a bipartite graph as shown in Fig. 7(b). Since $lcm\{PI_1, PI_2\} = 6$, we consider $\frac{6}{PI_1} = 3$ and $\frac{6}{PI_2} = 2$ packet arrivals of C_1 and C_2, respectively. Since each packet of C_2 has 2 bits, there are 7 vertices in V_l. In V_r, there are $B \times lcm\{PI_1, PI_2\} = 2 \times 6 = 12$ vertices. For example, bits $C_{2,2,1}$ and $C_{2,2,2}$ can be assigned to the sixth frame (of the current round) and the first frame (of the next round). The bold lines in Fig. 7(b) show one maximum-matching solution.

Theorem 2. *A scheduling problem has a feasible schedule if and only if its corresponding bipartite graph $G = (\{V_l, V_r\}, E)$ has a maximum matching with size of $|V_l|$.*

Proof. In the above, we show how to translate a scheduling problem to a maximum matching problem in a bipartite graph $G = (\{V_l, V_r\}, E)$. If there is a feasible solution for the scheduling problem, it means that each bit arrival is assigned its bit resource in the solution without violating the delay bound, where each bit arrival (resp., each bit resource) in the scheduling problem has a corresponding vertex in V_l (resp., V_r). Therefore, each bit resource assignment in the feasible solution can be mapped to an edge in E and all the bit arrivals being scheduled implies that the feasible solution can be mapped to a maximum bipartite matching in $G = (\{V_l, V_r\}, E)$ with size of $|V_l|$. This proves the *if* part.

On the contrary, if we can find a maximum matching in the corresponding bipartite graph G of a scheduling problem with size of $|V_l|$, it implies that each bit arrival in the scheduling problem during the period of L continuous frames is assigned a bit of resource without violating the delay bound constraint. Thus, the packet arrivals of all C_is are satisfactorily scheduled. This proves the *only if* part. □

5. Performance evaluation

To verify our result, we have simulated a BS-MS pair with multiple real-time connections by developing a simulator in C++. Unless otherwise stated, the following assumptions are made in our simulation. The number of connections n is ranged from 1 to 30. Each connection C_i has a data rate of 320–3200 bits/frame, delay bound of 50–1000 ms, and PI of 20–200 ms, where 320 is the minimum data rate, 3200 is the maximum data rate, 50 is the minimum delay bound, 1000 is the maximum delay bound, 20 is the minimum PI, 200 is the maximum PI of the MS. The maximum available resource per frame for the MS is $B = 20000$ bits and the length of an OFDM/OFDMA frame is set to 5 ms [18]. We consider three performance metrics: (i) *active ratio*: the ratio of active frames for the MS, (ii) *resource utilization*: the ratio of the amount of resource consumed by the MS to the total amount allocated to it, and (iii) *Fail ratio*: the ratio of failure to schedule the MS's sleep. We will compare our PSS-DB and PSS-PI against the PS scheme in [15] and an ideal *active ratio lower bound* (*ARL*), where ARL stands for the lowest active ratio for

the MS to support all given connections' traffics. We derive ARL by relaxing the delay bounds of connections as ∞, i.e., $D_i = \infty$, $i = 1..n$. Then, we can set the sleep cycle of the MS as $T_c \times \frac{lcm\{PI_i|i=1..n\}}{F}$ frames, where T_c is the minimum integer that makes (1) the sleep cycle be an integer and (2) the arrival data during the sleep cycle fill the frame up. Therefore, the listening window of the MS, T^L, can be derived as:

$$T^L = \frac{\sum_{i=1}^{n} \frac{T_c \times lcm\{PI_i|i=1..n\}}{PI_i} \times S_i}{B}.$$

Then, we can conduct ARL as follows:

$$ARL = \frac{\frac{\sum_{i=1}^{n} \frac{T_c \times lcm\{PI_i|i=1..n\}}{PI_i} \times S_i}{B}}{T_c \times lcm\{PI_i|i=1..n\}/F} = \sum_{i=1}^{n} \frac{S_i}{PI_i \times B} \times F. \quad (4)$$

Note that ARL provides only the value of active ratio, so we don't compare PSS-DB and PSS-PI against ARL in the resource utilization and fail ratio.

5.1. Effects of n

Fig. 8 shows the effect of n on the active ratio, resource utilization, and fail ratio by fixing $B = 80$ kbits/frame. Generally, as shown in Fig. 8(a), the active ratio increases as n increases. As n increases, we can see that, in the initial, when $n = 1$, three schemes, PS, PSS-DB, and PSS-PI, perform the same; but as n becomes larger, PSS-DB consumes the least active frames in three schemes and PS performs the worst because PS schedules the sleep of the MS by only considering the strictest delay bound among all connections while PSS-DB and PSS-PI can more accurately capture the required resource than the PS scheme by adapting each connection's sleeping cycle to its delay bound and PI, respectively, such that PSS-DB and PSS-PI can consume less active frames. As can be seen in Fig. 8(a), the difference of the active ratios between PSS-DB and PS increases as n increases when $n \leqslant 20$ (the same phenomenon can also be seen between PSS-PI and PS). The difference decreases when $n > 30$ because the network is becoming more and more saturated. In three schemes, PSS-DB has the smallest difference to ARL, which is an ideal lower bound for active ratio by assuming the delay bounds of connections as ∞. Fig. 8(b) show the resource utilization over different n. In general, the resource utilization decreases as n increases. PSS-DB always performs the best and no less than 88%, followed by the PSS-PI scheme, and PS performs the worst. This is because our approaches assign each connection a PSC such that the resource requirement can be more accurately captured compared to the PS scheme. Furthermore, PSS-DB shows that accumulating packets of a connection together according to the delay bound can help the resource utilization. Fig. 8(c) shows the schedule fail ratio of each scheme. We can see that PSS-DB always successfully schedules the MS into sleep (with fail ratio zero) while PSS-PI and PS start to have a probability to fail the sleep schedule when $n \geqslant 30$ and $n \geqslant 15$, respectively. Our PSS-DB and PSS-PI schemes show better performance than the PS scheme because PSS-DB and PSS-PI can more precisely capture the resource requirement of connections than PS such that less resource is required to be reserved (which has already been shown in Fig. 8(b)). Note that we have also done some experiments to see the packet drop rate because of the violation of delay bound for the three schemes. The results are all zero and show that our proposed schemes can guarantee the delay bound of packets like PS. Since the results are zero, we choose not to present these figures.

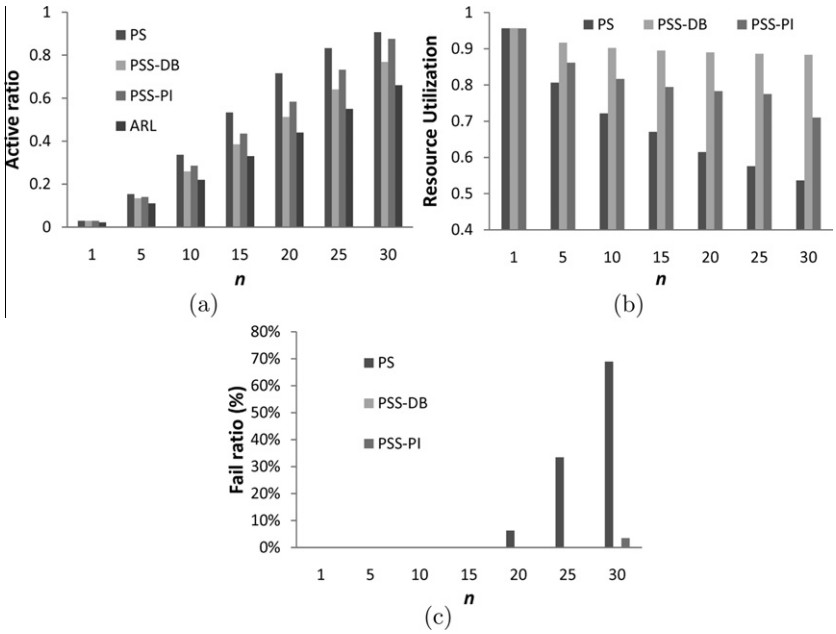

Fig. 8. Effects of n on (a) active ratio, (b) resource utilization, and (c) fail ratio with $B = 80$ kbits/frame.

5.2. Effects of maximum connection delay bound

We then investigate the effect of maximum connection delay bound on the active ratio, resource utilization, and fail ratio by fixing $n = 5$ and the minimum connection delay bound as 50 ms. As shown in Fig. 9(a), the active ratio decreases as the maximum connection delay bound increases. In three schemes, PSS-DB shows the best active ratio than PSS-PI and PS and has the smallest difference to ARL. Since ARL does not take the connection delay bound as a constraint, its active ratio is always the same over different delay ranges. Fig. 9(b) shows that PSS-DB performs the best resource utilization compared with PSS-PI and PS because it evaluates required resource of connections by delay bounds, causing the least resource waste. Generally, the resource utilization increases as the maximum connection delay bound increases for all three schemes. This is because the MS can have a longer sleeping cycle when the maximum connection delay bound increases. Fig. 9(c) shows that the PS scheme may fail to schedule the MS's sleep but this is not the case for PSS-DB and PSS-PI.

5.3. Effects of maximum connection packet inter-arrival time

In this experiment, we investigate the effect of maximum connection packet inter-arrival time (PI) on the active ratio, resource utilization, and fail ratio by fixing $n = 5$ and the minimum connection PI as 20 ms. Fig. 10(a) shows the active ratio increases when the maximum connection PI increases. When the data rate of a connection is fixed, a larger PI increases the packet size. This increases the penalty once a scheme cannot accurately capture the required resource of connections. This is why the PS scheme performs the worst of the three schemes. On the other hand, our PSS-DB and PSS-PI schemes both perform better than the PS scheme because they assign each connection a PSC according to its traffic characteristics such that the resource requirement can be more accurately captured and less active frames are consumed. As shown in Fig. 10(a), compared to the PS scheme, the active ratios of the PSS-DB and PSS-PI schemes improve almost 20% when the maximum connection PI is 650 ms. In Fig. 10(b), resource utilization decreases as the maximum connection PI increases. Despite the lower resource utilization our PSS-DB and PSS-PI schemes deliver a high utilization (over 86%) and converge. On the contrary, the resource utilization of the PS scheme keeps decreasing as the maximum connection PI increases. This is because a single PSC can not capture the traffic characteristics of connections. Fig. 10(c) shows that the PS scheme's fail ratio increases as the maximum connection PI increases while the PSS-DB and PSS-PI schemes have zero fail ratio. This shows our PSS-DB and PSS-PI schemes can more accurately capture connections' resource requirement again.

5.4. Effects of B

We consider two environments as follows. Environment *1* has five connections and each has a data rate of 320–3200 bits/frame, delay bound of 50–1000 ms, and PI of 20–200 ms. Environment 2 is for stress test, where we increase $n = 20$ and widen the range of PI to 20–500 ms. Fig. 11 shows the effect of B on the active ratio, resource utilization, and fail ratio under environments 1 and 2. Generally, as shown in Fig. 11(a) and (b), the active ratio decreases as B increases. In all three schemes, PSS-DB always performs the best and has the smallest difference to ARL, then is the PSS-PI scheme, and the PS scheme performs the worst. As shown in Fig. 11(a), our PSS-DB and PSS-PI schemes perform better than the PS scheme by 12.4–14.7% and 6.6–9.1%, respectively, in environment 1.

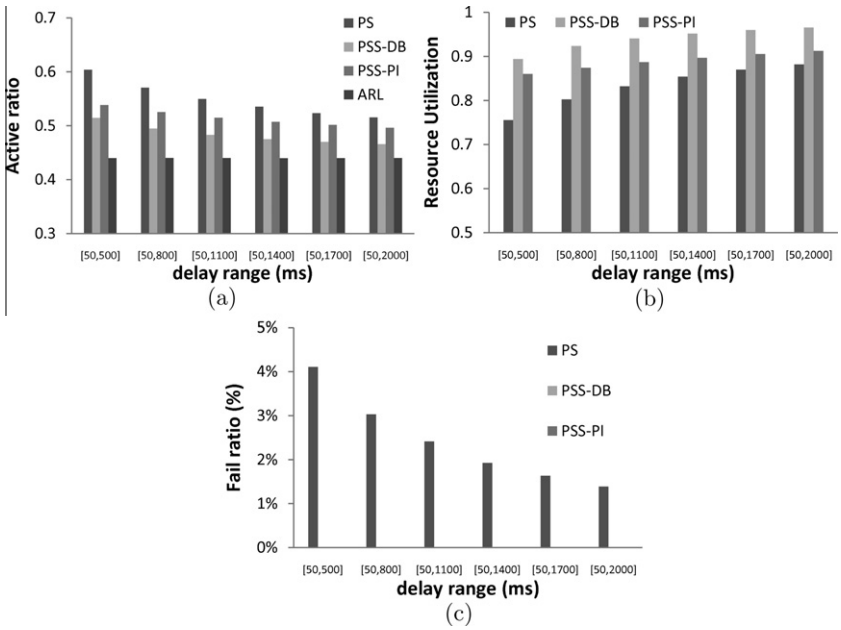

Fig. 9. Effects of maximum connection delay bound on (a) active ratio, (b) resource utilization, (c) fail ratio with $n = 5$.

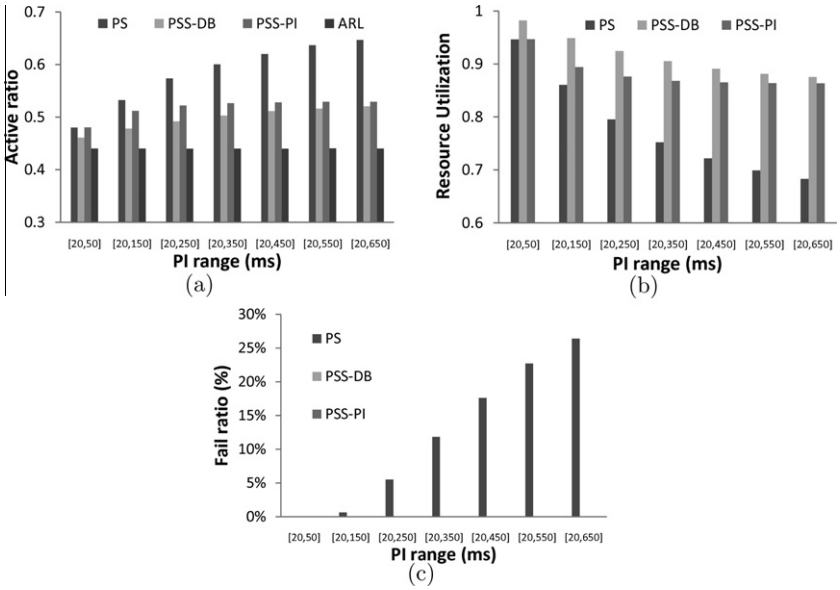

Fig. 10. Effects of maximum connection packet inter-arrival time on (a) active ratio, (b) resource utilization, and (c) fail ratio with $n = 5$.

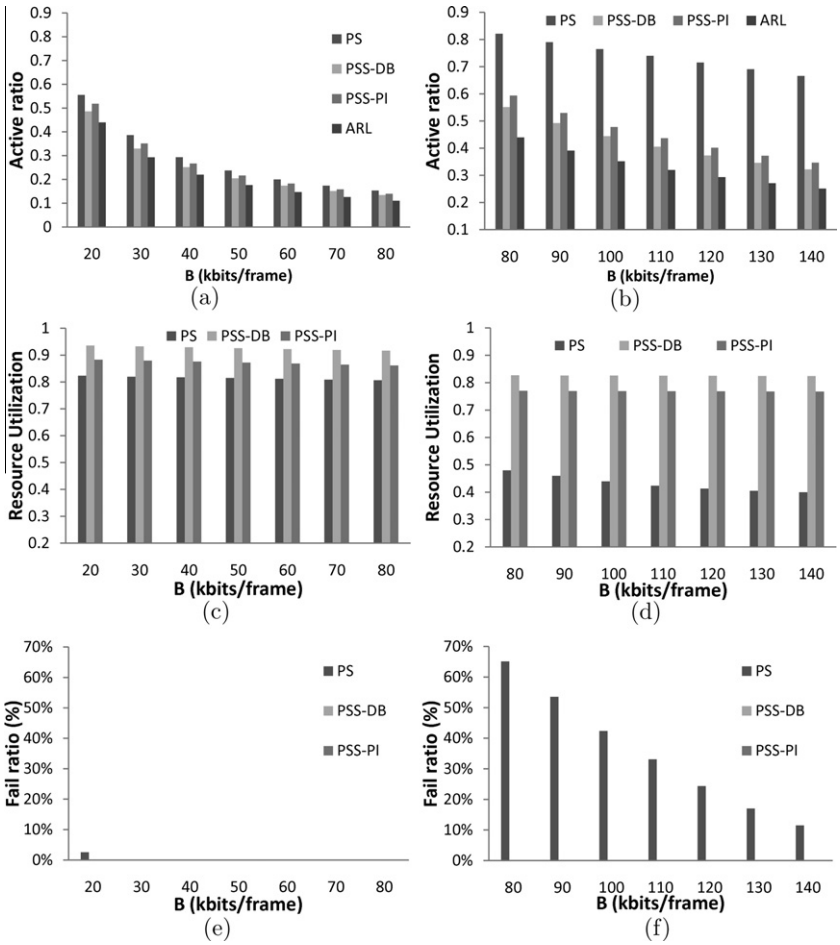

Fig. 11. Effects of B on active ratio, resource utilization, and fail ratio under environment 1 ((a), (c), and (e)) and 2 ((b), (d), and (f)).

In a stressful environment (environment 2 as shown in Fig. 11(b)), we can see that PS performs worse than that in Fig. 11(a) because it only considers the most strict delay bound of connections, thus the QoS characteristics of other connections are neglected. On the contrary, by assigning each connection a PSC, PSS-DB and PSS-PI still have low active ratios even in a stressful environment. As we can see in Fig. 11(b), our PSS-DB and PSS-PI schemes perform much better than PS by 32.9–51.6% and 27.7–48%, respectively. Fig. 11(c) and (d) show the resource utilization over different B. The resource utilizations of PSS-DB and PSS-PI are always more than 82% and 76%, respectively. This shows that our schemes can more precisely capture the resource requirement of connections than PS, which suffers severe resource utilization downgrade when the environment is stressful (only 39.5% resource utilization when $B = 140$ kbits/frame as shown in Fig. 11(d)). This also explains why the PSS-DB and PSS-PI schemes always have better active ratios than PS. Fig. 11(e) and (f) show the fail ratio of each scheme. In general, the fail ratio decreases as B increases. In environment 1, as shown in Fig. 11(e), since all three schemes have high resource utilization (over 80%), we can see they do not have any fail ratio except the PS scheme at $B = 20$ kbits/frame. On the other hand, the PS scheme has a severe fail ratio in environment 2 as shown in Fig. 11(f) because it cannot effectively capture the traffic characteristics of connections.

5.5. Active ratios by different candidate basic cycles (T_{basic})

Recall that T^*_{basic} is the basic cycle which induces the least active ratio for PSS-DB and PSS-PI. For each candidate T_{basic}, PSS-DB and PSS-PI adopt an exhausted search to find T^*_{basic}. However, this consumes time and computation power. To reduce this overhead, using the minimum connection delay bound as the direct basic cycle, i.e., $T_{basic} = \left\lfloor \frac{\min_{i=1..n}\{D_i\}}{F} \right\rfloor$, seems to be a possbile strategy because we can make T_1 and T_i, $i = 2..n$, be close to D_1 and D_i, respectively. Thus, the sleep schedule of each C_i can match its traffic characteristics. Fig. 12(a) and (b) show the performance evaluation for PSS-DB and PSS-PI by directly using the minimum connection delay bound as the basic cycle, respectively. In the experiment, the input parameters are $n = 20$ and each connection C_i has a data rate of 320–3200 bits/frame, delay bound of 50–1000 ms, and PI of 20–500 ms. As shown in Fig. 12(a), by using the minimum connection delay bound as the basic cycle, the PSS-DB performs in the middle of Optimal PSS-DB and Worst PSS-DB, where Optimal PSS-DB uses T^*_{basic} as the basic cycle and Worst PSS-DB always selects the worst candidate T_{basic}. As B increases, we can see the active ratio of the PSS-DB with the minimum connection delay bound as the basic cycle is closer and closer to Optimal PSS-DB. On the other hand, Fig. 12(b) shows that, using the minimum connection delay bound as the basic cycle, the PSS-PI performs close to Optimal PSS-PI. To summarize, if the cost of searching for T^*_{basic} is not a concern, we suggest to use T^*_{basic} to obtain the best performance for PSS-DB and PSS-PI. In fact, this exhaustive search is executed only when initialization. On the contrary, if time and computation power is a concern, using the minimum connection delay bound as the direct basic cycle is a compromise and as shown in our experiment, it will not be the worst case. Actually, as shown in Fig. 12, even using the minimum delay bound as the direct basic cycle, PSS-DB and PSS-PI still perform better than the PS scheme.

6. Conclusions

In this paper, we propose two per-flow sleep scheduling schemes, PSS-DB and PSS-PI, for IEEE 802.16 wireless networks, which guarantee the QoS of real-time connections. For each real-time connection, PSS-DB considers the delay bound to assign the sleeping cycle while PSS-PI uses the packet inter-arrival time to assign the sleeping cycle. Through these two multiple PSC solutions, required resource of the MS can be more accurately predicted than that of the single PSC solution; so the MS can sleep more and has a higher resource utilization. Also, the proposed schemes are compatible to the standard and easy to implement. Furthermore, both PSS-DB and PSS-PI perform better than the PS scheme regarding the active ratio, resource utilization, and fail ratio. We also prove that deciding whether a given scheduling problem is solvable can be reduced to a maximum matching problem, which can always be solved in polynomial time. In this work, we consider per-flow sleeping scheduling for one MS. For the case of multiple MSs, the operations of our schemes are not changed. But, the number of MSs influences the resource which could be assigned to each MS. Since the network bandwidth is fixed, the maximum available resource per frame B decreases when the number of MSs increases; B increases

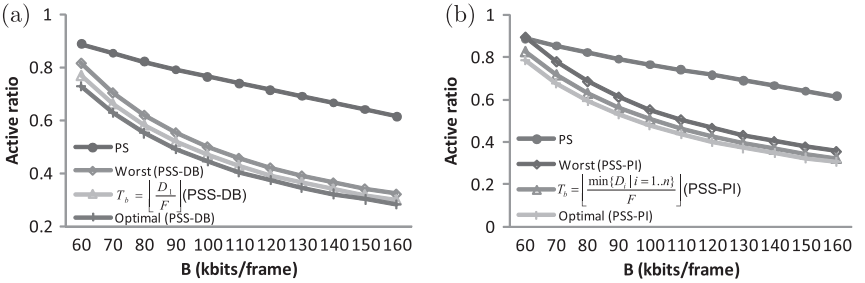

Fig. 12. Active ratio by using different T_{basic}: (a) PSS-DB and (b) PSS-PI with $n = 20$, data rate = 320–3200 bits/frame, delay bound = 50–1000 ms, and PI = 20–500 ms.

when the number of MSs decreases. As shown in our experiment (Section 5.4), the active ratio of an MS increases as its B decreases. On the other hand, for the system, when the number of MSs increases (This means that there is more data traffic), the utilization of system resource increases. So, the system resource utilization benefits when the number of MSs increases. In this work, we use an ideal lower bound, ARL, to evaluate the performance of our schemes. As our future work, we will try to develop a mathematical model for analyzing the performance of the proposed schemes and find the real optimum to evaluate our schemes.

Acknowledgment

Y.-C. Tseng's research is co-sponsored by MoE ATU Plan, by NSC Grants 97-3114-E-009-001, 97-2221-E-009-142-MY3, 98-2219-E-009-019, 98-2219-E-009-005, and 99-2218-E-009-005, by ITRI, Taiwan, by III, Taiwan, by D-Link, and by Intel. S.-L. Wu's research is supported by the National Science Council, ROC, under Grant NSC99-2221-E-182-039, and the High Speed Intelligent Communication (HSIC) Research Center of Chang Gung University.

References

[1] IEEE Std 802.16-2009, IEEE Standard for Local and metropolitan area networks. Part 16: Air Interface for Broadband Wireless Access Systems, May 2009.
[2] J.A. Stine, G.D. Veciana, Improving energy efficiency of centrally controlled wireless data Networks, ACM/Baltzer Wireless Networks 8 (6) (2002) 681–700.
[3] M. Anand, E.B. Nightingale, J. Flinn, Self-tuning wireless network power management, ACM/Baltzer Wireless Networks 11 (4) (2005) 451–469.
[4] F. Zhang, T.C. Todd, D. Zhao, V. Kezys, Power saving access points for IEEE 802.11 wireless network infrastructure, IEEE Transactions on Mobile Computing 5 (2) (2006) 144–156.
[5] Y. Xiao, Energy saving mechanism in the IEEE 802.16 e wireless MAN, IEEE Communications Letters 9 (7) (2005) 595–597.
[6] Y. Zhang, M. Fujise, Energy management in the IEEE 802.16 e MAC, IEEE Communications Letters 10 (4) (2006) 311–313.
[7] K. Han, S. Choi, Performance analysis of sleep mode operation in IEEE 802.16 e mobile broadband wireless access systems, in: Proceedings of IEEE 63rd Vehicular Technology Conference (VTC'06-Spring), vol. 3, May 2006, pp. 1141–1145.
[8] Y. Zhang, Performance modeling of energy management mechanism in IEEE 802.16 e mobile WiMAX, in: Proceedings of IEEE Wireless Communications and Networking Conference (WCNC'07), March 2007, pp. 3205–3209.
[9] J. Xiao, S. Zou, B. Ren, S. Cheng, An enhanced energy saving mechanism in IEEE 802.16 e, in: Proceedings of IEEE GLOBECOM'06, November 2006.
[10] S. Cho, Y. Kim, Improving power savings by using adaptive initial-sleep window in IEEE802.16 e, in: Proceedings of IEEE VTC'07-Spring, April 2007, pp. 1321–1325.
[11] F. Xu, W. Zhong, Z. Zhou, A novel adaptive energy saving mode in IEEE 802.16 e System, in: Proceedings of Military Communications Conference (MILCOM'06), October 2006.
[12] J.-R. Lee, D.-H. Cho, Performance Evaluation of Energy-Saving Mechanism Based on Probabilistic Sleep Interval Decision Algorithm in IEEE 802.16 e, IEEE Transactions on Vehicular Technology 56 (4) (2007) 1773–1780.
[13] M.-G. Kim, J.-Y. Choi, M. Kang, Adaptive power saving mechanism considering the request period of each initiation of awakening in the IEEE 802.16 e system, IEEE Communications Letters 12 (2) (2008) 106–108.
[14] T.-C. Chen, J.-C. Chen, Y.-Y. Chen, Maximizing unavailability interval for energy saving in IEEE 802.16 e wireless MANs, IEEE Transactions on Mobile Computing 8 (4) (2009) 475–487.
[15] S.-L. Tsao, Y.-L. Chen, Energy-efficient packet scheduling algorithms for real-time communications in a mobile WiMAX system, Computer Communications 31 (10) (2008) 2350–2359.
[16] H.-L. Tseng, Y.-P. Hsu, C.-H. Hsu, P.-H. Tseng, K.-T. Feng, A maximal power-conserving scheduling algorithm for broadband wireless networks, in: Proceedings of IEEE WCNC'08, March 2008, pp. 1877–1882.
[17] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, 2001.
[18] H.S. Kim, S. Yang, Tiny MAP: an efficient MAP in IEEE 802.16/WiMAX broadband wireless access systems, Computer Communications 30 (9) (2007) 2122–2128.

Jen-Jee Chen received his BS and MS degrees in Computer Science and Information Engineering from the National Chiao Tung University, Taiwan, in 2001 and 2003, respectively. He was a Visiting Scholar at the University of Illinois at Urbana-Champaign during the 2007–2008 academic year. Then, he obtained his Ph.D. in Computer Science from the National Chiao Tung University, Taiwan, in October of 2009. He was a post-doctoral research fellow (2010-2011) at the Department of Electrical Engineering, National Chiao Tung University, Taiwan. He is Assistant Professor (2011-present) at the Department of Electrical Engineering, National University of Tainan, Taiwan. His research interests include wireless communications and networks, personal communication networks, mobile computing, cross-layer design, and cloud computing. Dr. Chen is a member of the IEEE and the Phi Tau Phi Society.

Shih-Lin Wu received the B.S. degree in Computer Science from Tamkang University, Taiwan, in June 1987 and the Ph.D. degree in Computer Science and Information Engineering from National Central University, Taiwan, in May 2001. He was an Assistant Professor at Chang Gung University (2001–2007). He is Associate Professor (2007∼preset) and Chairman (2007-present) at the Department of Computer Science and Information Engineering, Chang Gung University. His current research interests include mobile communications, wireless networks, and distributed robotics. He serves as a member of editor board of Telecommunication Systems, Journal of Positioning and ISRN Communications. He was a Guest Editor of International Journal of Pervasive Computing and Communications 2007, a Program Chair of Mobile Computing 2005, a Co-Chair of Workshop on IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing 2006, a Program Chair of International Workshop on Data Management in Ad Hoc and Pervasive Computing 2009, a Co-Chair of International High Speed Intelligent Communication 2009, and a Co-Chair of International Symposium on Bioengineering 2011. Several of his papers have been chosen as Selected/Distinguished Papers in international/local conferences. Dr. Wu is a member of the IEEE and the Phi Tau Phi Society.

Shiou-Wen Wang received her BS degree in Information Engineering and Computer Science from the Feng Chia University, Taiwan, in 2006. Then, she obtained her MS degree in Computer Science and Information Engineering from the Chang Gung University, Taiwan, in 2009. Her research interests include wireless communications and networks and mobile computing. She has been with Panteck Technology Corp., Taipei, Taiwan, as a software engineer since 2009.

Yu-Chee Tseng got his Ph.D. in Computer and Information Science from the Ohio State University in January of 1994. He is/was Professor (2000-present), Chairman (2005–2009), and Associate Dean (2007-present), Department of Computer Science, National Chiao-Tung University, Taiwan, and Chair Professor, Chung Yuan Christian University (2006–2010).

Dr. Tseng received Outstanding Research Award (National Science Council, 2001, 2003, and 2009), Best Paper Award (Internationall Conference on Parallel Processing, 2003), Elite I. T. Award (2004), and Distinguished Alumnus Award (Ohio State University, 2005), and Y. Z. Hsu Scientific Paper Award (2009). His research interests include mobile computing, wireless communication, and parallel and distributed computing.

Dr. Tseng serves/served on the editorial boards for Telecommunication Systems (2005-present), IEEE Trans. on Vehicular Technology (2005–2009), IEEE Trans. on Mobile Computing (2006-present), and IEEE Trans. on Parallel and Distributed Systems (2008-present).