Low-Complexity Class-Based Scheduling Algorithm for Scheduled Automatic Power-Save Delivery for Wireless LANs

Tsern-Huei Lee, *Senior Member*, *IEEE*, and Jing-Rong Hsieh, *Member*, *IEEE*

Abstract—Power saving is an important issue when integrating the wireless LAN technology into mobile devices. Besides Quality of Service (QoS) guarantee, the IEEE 802.11e introduces an architecture called Scheduled Automatic Power-Save Delivery (S-APSD) aiming at delivering buffered frames to power save stations. In S-APSD, the Access Point (AP) schedules the Service Period (SP) of stations. To increase power efficiency, SPs should be scheduled to minimize the chance of overlapping. In a recent paper, an algorithm named Overlapping Aware S-APSD (OAS-APSD) was proposed to find the wake-up time schedule for a new Traffic Stream (TS) to minimize the chance of SP overlapping. The combination of OAS-APSD and HCF Controlled Channel Access (HCCA) was proved to outperform 802.11 Power Save Mode (PSM) with Enhanced Distributed Channel Access (EDCA) in power saving efficiency and QoS support. However, the OAS-APSD algorithm requires high online computational complexity which could make it infeasible for real systems. Without harming the optimality, this paper presents an efficient algorithm with much less complexity by exploiting the periodicity of service schedule. Because of largely reduced online computational complexity, the proposed algorithm is much more feasible than OAS-APSD.

Index Terms—Wireless LAN, scheduling, power saving

1 Introduction

The IEEE 802.11 [1] wireless LAN has been widely spread due to its low cost and easy installation. One can easily find wireless LAN hotspots in most places of a modern city such as office, campus, café, or even on the street. Therefore, more and more mobile devices include wireless LAN functionality as a method for accessing the Internet or sharing files and multimedia data between peer devices. As the hardware performance of the mobile devices is greatly improved and many useful features such as location-based service are introduced, it is more likely for people to access the Internet anytime and anywhere through their mobile devices. However, to provide Quality of Service (QoS) guarantee while prolonging the usage time of mobile devices, several challenges need to be settled.

To cope with QoS support, IEEE 802.11e standard [2], an enhancement of 802.11, defines a QoS-aware coordination function called Hybrid Coordination Function (HCF). This function consists of two channel access mechanisms. One is contention-based Enhanced Distributed Channel Access (EDCA) and the other is contention-free HCF Controlled Channel Access (HCCA). Because of the contention-free nature, HCCA can provide much better QoS guarantee than EDCA. EDCA can be used only during contention period while HCCA can be used in both contention period and contention-free period. Interested readers can find an overview of the 802.11e QoS enhancements in [3]. The IEEE 802.11e and other amendments finished before the year 2005 had been merged with the 1999 version of the 802.11 standard and the currently published specification is the IEEE 802.11-2007 [16], [17].

Regarding power saving for wireless LAN, most previous works consider ad hoc scenarios because devices in an infrastructure system are usually connected to a power supply or equipped with long-life batteries. The situation is changed for multimode mobile devices because their small size severely limits the battery size. The IEEE 802.11 standard provides a power management mechanism at the MAC layer, known as Power Save Mode (PSM). When using PSM, a station (STA) sleeps and wakes up regularly to listen to beacons transmitted by Access Point (AP). It is assigned an Association ID (AID) during the association process. If its AID is indicated in the Traffic Indication Map of the beacon, meaning that there are data buffered at AP, the STA remains awake and tries to retrieve the data by sending PS-Poll frames. AP will set the More Data bit in the data frame it sends to the STA if there are more frames buffered for the STA. The STA enters the Doze state only when all its data are retrieved. Thus, the time it spends to listen to the channel is reduced. Obviously, under the PSM mechanism, delay of downlink frames depends on the STA's listening interval which is multiples of the beacon interval. This may not be acceptable for real-time applications. Quantitative evaluation for combinations of PSM and 802.11e can be found in [5].

To provide QoS support for unicast traffic and achieve power saving, the 802.11e standard includes an extension of the PSM mechanism, called Automatic Power Save Delivery (APSD). Two different APSD modes were defined.

• *The authors are with the Institute of Communication Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. E-mail: {tlee, jingrong}@banyan.cm.nctu.edu.tw.*

Manuscript received 2 Dec. 2009; revised 10 Jan. 2011; accepted 30 Dec. 2011; published online 8 May 2012.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2009-12-0524. Digital Object Identifier no. 10.1109/TMC.2012.114.

Unscheduled APSD (U-APSD) is a distributed mechanism where STAs decide when to awake to retrieve their data buffered at QoS AP (QAP). Scheduled APSD (S-APSD) is a centralized mechanism where QAP determines the wake-up periods and wake-up times for STAs. There exist some famous scheduling algorithms such as Rate Monotonic [11] and Earliest Deadline First [15] that were designed for real-time OS and server/switch which handle delay-sensitive traffic. Unfortunately, their scheduling results are in general not periodic and, therefore, not suitable for power saving purpose. In [4], it was shown that the S-APSD mechanism combined with HCCA provide excellent QoS support and power saving. However, the scheduling algorithm proposed in [4] requires high computational complexity which can make it infeasible for real systems.

The purpose of this paper is to present a low complexity scheduling algorithm which achieves QoS support and power saving simultaneously. The scheduling criterion is slightly different from that adopted in [4]. To reduce online computational complexity, we classify Traffic Streams (TS) based on their wake-up periods and store necessary information to allow fast scheduling. According to simulation results, our proposed scheduling algorithm obtains almost the same energy consumption as that obtained by the scheduling algorithm presented in [4], with much smaller decision time.

The rest of this paper is organized as follows: in Section 2, we review the S-APSD mechanism and the scheduling algorithm proposed in [4]. Section 3 describes the idea of our proposed scheduling algorithm. The idea is further extended to a system with multiple classes of TSs in Section 4. In Section 5, we compare our proposed algorithms with the existing work [4] in terms of computational complexity and energy consumption. Finally, we draw conclusion in Section 6.

2 Related Works

2.1 Scheduled Automatic Power-Save Delivery (S-APSD)

To support QoS while dealing with power saving issue, U-APSD and S-APSD architectures are defined in IEEE 802.11e [2], [4]. They avoid the necessity of PS-Poll frames when retrieving downlink frames. The U-APSD requires STAs to contend for the channel to transmit an uplink data frame or null frame to trigger the delivery of the buffered downlink frames from QAP. The QAP should use EDCA access method when selecting the U-APSD scheme. On the other hand, the S-APSD lays the burden of channel coordination on the QAP to calculate the schedule and announce it to STAs. When S-APSD is used, depending on whether the usage is for an Access Category (AC) or for a TS, EDCA, or HCCA is chosen as access policy, respectively.

For the S-APSD, the STA first communicates with the QAP via Add Traffic Stream (ADDTS) request frame setting both APSD and Schedule subfields in the TS info field before getting admitted. If the requested service can be satisfied, the QAP will notify the STA of the schedule including the Service Start Time (SST) and the negotiated Service Interval (SI) in the schedule element. As shown in Fig. 1, both the SST and SI fields are four octets and carry

Octets: 1	1	2	4	4	2
Element ID	Length	Schedule Info	Service Start Time	Service Interval	Specification Interval

Fig. 1. The schedule element of S-APSD.

time values in microseconds. Note that although the services are delivered periodically in S-APSD, the conveyed sources of applications are not necessarily to be periodic or constant-bit-rate. In APSD, the contiguous time that an STA stays awake to receive the buffered frames from QAP is defined as Service Period (SP). STAs using S-APSD should automatically switch to the Awake state at the scheduled starting time of each SP defined by

$$SST + m \times SI, \text{ where } m \geq 0, m \in N. \quad (1)$$

Then, they fall back to sleep till receiving the frames with the End Of Service Period (EOSP) flag being set. The service schedule can be updated after negotiation between the QAP and STA finishes. To maintain QoS guarantee, the new SST should fall into the region between the minimum SI and maximum SI after the beginning of the previous SP. Compared with the 802.11 PSM, besides QoS support, the S-APSD can also reduce signaling loads such as PS-Poll. Moreover, the number of collisions can be decreased as well.

2.2 Overlapping Aware S-APSD (OAS-APSD) [4]

Although IEEE 802.11e defines the architecture of S-APSD, its specific implementation is left as an open issue. For the scenario with multiple STAs which wake up periodically to retrieve their buffered data, the overlapping of SPs is the major source that wastes their energy because they may be awake for durations longer than their transmissions. Since the medium is shared among STAs, an STA may spend energy on overhearing the transmissions between the QAP and other STAs before returning to sleep.

To reduce the chance of SP overlapping, as described in [4], there could be two scheduling approaches to schedule the starting time of SPs. One is contiguous scheduling, which means that the scheduled SPs should be placed one after another. It has the advantage of simplifying the process to determine the SST for the schedule of a new TS. However, it often requires the SIs to be altered to satisfy certain constraint so that contiguous scheduling is possible. A necessary and sufficient condition for a group of periodic tasks defined by SIs and Transmission Opportunities (TXOPs) to be scheduled contiguously by an Equal-spacing-based Rate Monotonic algorithm was derived in [6]. Altering SIs may shorten the sleeping time of STAs and, as a result, cause more energy consumption to retrieve the same amount of data buffered at QAP. Besides, contiguous scheduling is only suitable for constant-bit-rate traffic. For variable-bit-rate traffic, it is often a difficult task to determine the duration of an SP to achieve high efficiency in both energy consumption and bandwidth utilization. In [10], Hsieh et al. formulated the problem as one which optimizes energy consumption given an upper bound of bandwidth loss. However, it was assumed that the

distribution of traffic arrival of each TS is stationary and known. The assumption may not be realistic and, even if it is acceptable, the huge computational complexity prohibits the optimal solution from being adopted.

In [4], a noncontiguous scheduling algorithm called Overlapping Aware S-APSD was proposed. The OAS-APSD algorithm aims at finding the SST of a new TS which achieves the least probability of SP overlapping. The pseudocode of the OAS-APSD algorithm is shown below. To be concise, we use scheduled instants to represent the scheduled starting time of SPs. The Scheduled Events (*SEs*) in the OAS-APSD algorithm refer to the scheduled instants known to the QAP, for example, Beacons with period BI and already scheduled SPs with period SIs for TSs. In this algorithm, SI_{new} represents the SI of the new TS to be scheduled.

The OAS-APSD algorithm [4].

SST **to be determined given a specific** SI_{new}
$N, SST, SST_{temp}, dist_{avg}, temp\_dist_{avg}, max\_dist_{min} \leftarrow 0$
$temp\_dist_{min} \leftarrow BI$
Create empty list of $SEs \rightarrow ListSE$
Compute LCM **considering** $All\ SIs$ **plus** $BI \rightarrow LCM$
for $\forall SEs \in [t_{current}, t_{current} + LCM]$ **do**
  Insertion in $ListSE$ **of** SEs
end for
while $SST_{temp} < SI_{new}$ **do**
  while $SST_{temp} + SI_{new} \times N < LCM$ **do**
    Find $prev\_SE$ **and** $next\_SE$ **in** $ListSE$
      $dist_{next\_SE} \leftarrow next\_SE - (SST_{temp} + SI_{new} \times N)$
      $dist_{prev\_SE} \leftarrow SST_{temp} + SI_{new} \times N - prev\_SE$
    Insertion in $distances\_SST_{temp}$ **of** $dist_{next\_SE}$ **and** $dist_{prev\_SE}$
    $N \leftarrow N + 1$
  end while
  $temp\_dist_{min} \leftarrow$ **Minimum of** $distances\_SST_{temp}$
  $temp\_dist_{avg} \leftarrow$ **Average of** $distances\_SST_{temp}$
  if $temp\_dist_{min} > max\_dist_{min}$ **then**
    $max\_dist_{min} \leftarrow temp\_dist_{min}$
    $dist_{avg} \leftarrow temp\_dist_{avg}$
    $SST \leftarrow SST_{temp}$
  else if $temp\_dist_{min} = max\_dist_{min}$ **then**
    if $temp\_dist_{avg} > dist_{avg}$ **then**
      $dist_{avg} \leftarrow temp\_dist_{avg}$
      $SST \leftarrow SST_{temp}$
    else if $temp\_dist_{avg} = dist_{avg}$ **then**
      $SST \leftarrow random(SST, SST_{temp})$
    end if
  end if
  $SST_{temp} \leftarrow SST_{temp} + precision$
end while

The basic idea of the OAS-APSD algorithm is to find the optimal SST of the new TS in an interval [0, SI_{new} -1] which achieves the maximum among minimum relative distances between SPs of the new TS and existing scheduled events. Here, the relative distances between SPs of the new TS and existing scheduled events are defined as the distances from the starting time of every SP of the new TS to its closest previous and next existing scheduled events, as illustrated in Fig. 2. Note that there are only a finite number of possibilities for the SST because there is a maximum precision used by the 802.11 specification. According to [1], [12], the timing synchronization function of each STA is based on a 1-MHz clock and thus the time ticks in microseconds. As a result, the system is actually slotted with slot size or maximum precision equal to multiple of one microsecond. Without loss of generality, the duration of the maximum precision is normalized to 1. As a result, the SIs and the scheduled events are integers.

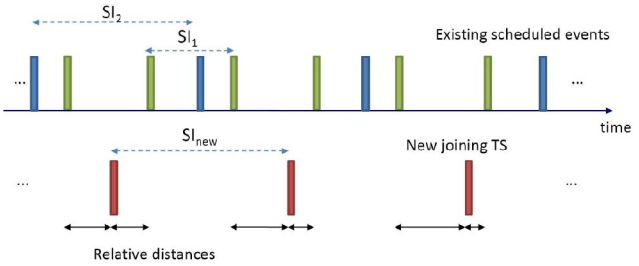

Fig. 2. Illustration of relative distances between scheduled events.

To determine the optimal SST, relative distances are calculated for an interval of duration *LCM*, the least common multiple of the SIs, including SI_{new}. If there is a tie, then the one with maximum average relative distance is selected. In case there is still a tie based, on average, relative distance, it is broken arbitrarily. Clearly, the computational complexity of the OAS-APSD algorithm is large for large values of *LCM*. This can make the algorithm infeasible for real systems.

Note that the scheduled instants for a specific TS can be represented as a sequence, say,

$$\boldsymbol{X} = \{x_m\}_{m=-\infty}^{\infty}, \text{ such that } x_m = x_{m-1} + p, \tag{2}$$

where p, called period, is the SI of the TS. The origin can be chosen arbitrarily. Consequently, the SST of the TS corresponds to some x_i and the STA to which the TS is attached wakes up periodically at time instants x_j for all $j \geq i$.

3 The Proposed Low-Complexity Scheduling Algorithm

3.1 Basic Idea

In this section, we present the basic idea of determining the optimal SST for a new TS. Consider the simplest case of scheduling the SPs for the first TS with period p. Let $\boldsymbol{X} = \{x_m\}_{m=-\infty}^{\infty}$ be the scheduled instants for the first TS. It is clear that any sequence of period p is optimal. For simplicity, we choose $x_m = p \times m$ for all m.

Consider now the case where an existing TS with period p had been scheduled and a new TS is to be scheduled. Assume that the period of the new TS to be scheduled is q. Let $\boldsymbol{Y} = \{y_m\}_{m=-\infty}^{\infty}$ be a periodic sequence of period q such that $y_m = q \times m$ for all m. Further, let

$$\boldsymbol{Y} + k = \{y_m + k\}_{m=-\infty}^{\infty} \tag{3}$$

be a shifted version of $\boldsymbol{Y}$. We call k the offset of $\boldsymbol{Y} + k$ with respect to $\boldsymbol{Y}$. It is clear that $\boldsymbol{Y} + k$ is periodic in k with period q.

Define the distance between sequences $\boldsymbol{X}$ and $\boldsymbol{Y}+k$ as

$$d(\boldsymbol{X},\boldsymbol{Y}+k)=\min_{-\infty\le l,m\le\infty}|y_l+k-x_m|. \tag{4}$$

It is not hard to see that

$$d(\boldsymbol{X},\boldsymbol{Y}+k)=d(\boldsymbol{Y}+k,\boldsymbol{X})=d(\boldsymbol{X}-k,\boldsymbol{Y}), \tag{5}$$

and

$$d(\boldsymbol{X},\boldsymbol{Y}+0)=d(\boldsymbol{X},\boldsymbol{Y})=0. \tag{6}$$

Define

$$D(\boldsymbol{X},\boldsymbol{Y})=\max_k\{d(\boldsymbol{X},\boldsymbol{Y}+k)\}. \tag{7}$$

Our goal is to find k^*, the optimal value of k which satisfies

$$k^*=\arg\max_k d(\boldsymbol{X},\boldsymbol{Y}+k). \tag{8}$$

Once k^* is obtained, the scheduled instants of the new TS is $\boldsymbol{Y}+k^*$ and the SST can be determined based on current time. Let $G=\gcd(p,q)$ and $L=\mathrm{lcm}\{p,q\}$ be, respectively, the greatest common divisor and the least common multiple of p and q. We prove in Theorem 1 a property of $d(\boldsymbol{X},\boldsymbol{Y}+k)$.

Theorem 1. *$d(\boldsymbol{X},\boldsymbol{Y}+k)$ is periodic in k with period G.*

Proof. It is clear that $d(\boldsymbol{X},\boldsymbol{Y}+k)$ is periodic in k because $d(\boldsymbol{X},\boldsymbol{Y}+q+k)=d(\boldsymbol{X},\boldsymbol{Y}+k)$. Let n be its period. We shall prove that $n|G$ (i.e., n divides G) and $G|n$.

According to the euclidean algorithm [7], there exist integers a and b such that

$$G=a\times p+b\times q \text{ or } G+(-b)\times q=a\times p, \tag{9}$$

which implies one of the scheduled instants of $\boldsymbol{Y}+G$ coincides with some scheduled instant of $\boldsymbol{X}$. Consequently, we have $d(\boldsymbol{X},\boldsymbol{Y}+G+k)=d(\boldsymbol{X},\boldsymbol{Y}+k)$, which implies $n|G$.

Conversely, since n is the period of $d(\boldsymbol{X},\boldsymbol{Y}+k)$, we have $d(\boldsymbol{X},\boldsymbol{Y}+0)=d(\boldsymbol{X},\boldsymbol{Y}+n)$, which implies there must exist integers s and t such that

$$n+s\cdot q=t\cdot p \text{ or } n=(-s)\cdot q+t\cdot p. \tag{10}$$

As a result, it holds that $G|n$ because $G|p$ and $G|q$. This completes the proof of Theorem 1. □

A consequence of Theorem 1 is that k^* can be chosen to satisfy $0\le k^*\le G-1$. Note that to compute $d(\boldsymbol{X},\boldsymbol{Y}+k)$, $0\le k\le G-1$, we need only consider finite partial sequences of $\boldsymbol{X}$ and $\boldsymbol{Y}+k$ because the same situation repeats every L slots. Two cases are analyzed separately below.

Case 1. $q\ge p$.

For $q\ge p$, we need only consider $\{x_m\}_{m=0}^{L/p-1}$ and $\{y_m+k\}_{m=0}^{L/q-1}$. Let

$$f_m(k)=\lfloor(q\cdot m+k)/p\rfloor, \tag{11}$$

where $\lfloor x\rfloor$ represents the largest integer smaller than or equal to x. Define

$$\begin{aligned}a_m(k)=\min\{&q\cdot m+k-p\cdot f_m(k),\\ &p\cdot[f_m(k)+1]-(q\cdot m+k)\},\end{aligned} \tag{12}$$

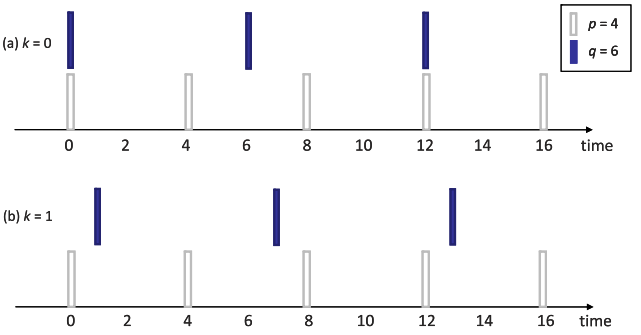

Fig. 3. S-APSD scheduling example.

as the shorter distance of y_m+k to the two closest neighboring x_m's. We have

$$d(\boldsymbol{X},\boldsymbol{Y}+k)=\min_{0\le m\le L/q-1}\{a_m(k)\}. \tag{13}$$

Figs. 3a and 3b show an example for $p=4$ and $q=6$. For this example, we have $G=2$ and, therefore, we need only compute $d(\boldsymbol{X},\boldsymbol{Y}+0)$ and $d(\boldsymbol{X},\boldsymbol{Y}+1)$. One can easily verify that $a_0(0)=\min\{0,4\}=0$, $a_1(0)=\min\{2,2\}=2$, $d(\boldsymbol{X},\boldsymbol{Y}+0)=\min\{0,2\}=0$; and $a_0(1)=\min\{1,3\}=1$, $a_1(1)=\min\{3,1\}=1$, $d(\boldsymbol{X},\boldsymbol{Y}+1)=\min\{1,1\}=1$. As a consequence, we have $D(\boldsymbol{X},\boldsymbol{Y})=\max\{d(\boldsymbol{X},\boldsymbol{Y}+0),d(\boldsymbol{X},\boldsymbol{Y}+1)\}=1$ and $k^*=1$.

Case 2. $q<p$.

For $q<p$, one can compute $a_m(k)$ for $0\le m\le L/q-1$ and then determine $d(\boldsymbol{X},\boldsymbol{Y}+k)=\min_{0\le m\le L/q-1}\{a_m(k)\}$. Alternatively, one can change the roles of p and q and apply the procedure performed for Case 1. Note that $d(\boldsymbol{X},\boldsymbol{Y}+k)=d(\boldsymbol{X}-k,\boldsymbol{Y})=d(\boldsymbol{X}+G-k,\boldsymbol{Y})$ implies the desired results can be obtained by interchanging the roles of p and q. By doing so, the complexity of computing $d(\boldsymbol{X},\boldsymbol{Y}+k)$ is reduced because fewer $a_m(k)$'s (L/p *versus* L/q) are calculated.

To summarize, in order to determine $d(\boldsymbol{X},\boldsymbol{Y}+k)$, we need to compute $L/\max\{p,q\}$ $a_m(k)'s$ and then pick the smallest one. Since there are G different values for variable k, the complexity of determining $d(\boldsymbol{X},\boldsymbol{Y}+k)$, $0\le k\le G-1$, is $O(G\cdot L/\max\{p,q\})=O(\min\{p,q\})$ multiplications and divisions. The following algorithm eliminates all multiplications and divisions. The algorithm requires roughly $4\times\min\{p,q\}$ comparisons.

Algorithm ***for*** *computing $D(\boldsymbol{X},\boldsymbol{Y})$ and k^* assuming that $q\ge p$.*

```
R = q mod p
D, k* ← 0                    /* D stores the value of D(X,Y).*/
k ← 1
while k < G
    m ← 0
    S ← k    /* S stores the value of q·m+k−p·f_m(k). */
    relative_dist ← min{S, p − S}
    min_relative_dist ← relative_dist
    while m ≤ L/q − 1
        S ← S + R
        if S > p
            S ← S − p
        end if
        relative_dist ← min{S, p − S}
```

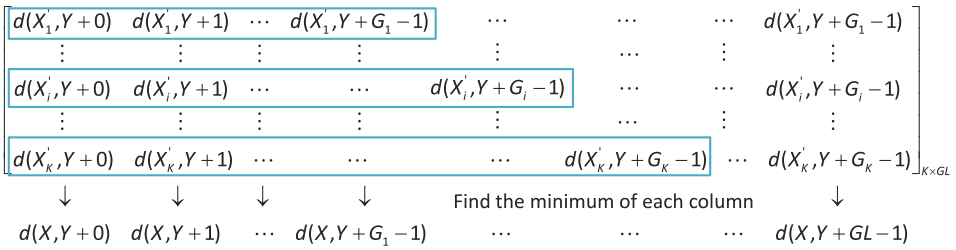

Fig. 4. The scheduling matrix.

$min\_relative\_dist \leftarrow \min\{relative\_dist, min\_relative\_dist\}$
$m \leftarrow m + 1$
end while
if $D < min\_relative\_dist$
$D \leftarrow min\_relative\_dist$
$k^* \leftarrow k$
else if $D = min\_relative\_dist$
$k^* \leftarrow random(k^*, k)$
end if
$k \leftarrow k + 1$
end while

3.2 Generalization to K Existing TSs

Let us now extend the results to $K \geq 2$ existing TSs when a new TS is to be scheduled. Assume that the period of the ith existing TS is p_i and the period of the TS to be scheduled is q. Let $\boldsymbol{X}_i = \{x_{i,m}\}_{m=-\infty}^{\infty}$, $1 \leq i \leq K$, be a sequence of period p_i such that $x_{i,m} = p_i \cdot m$ for all m. Further, let

$$\boldsymbol{X}_i' = \boldsymbol{X}_i + O_i, \tag{14}$$

be the scheduled instants of the ith existing TS. We shall use $\boldsymbol{X}_1$ as reference and, therefore, assign $O_1 = 0$. Consequently, we have $\boldsymbol{X}_1' = \boldsymbol{X}_1$ and

$$O_i = x_{i,0} - x_{1,0}, \tag{15}$$

represents the offset of $\boldsymbol{X}_i'$ with respect to $\boldsymbol{X}_1'$. According to the results obtained in Section 3.1, it holds that $0 \leq O_2 \leq p_2 - 1$. We shall prove that O_i satisfies $0 \leq O_i \leq p_i - 1$ for all i. Let $G_i = \gcd(p_i, q)$, $1 \leq i \leq K$, and

$$GL = \text{lcm}\{G_1, G_2, \ldots, G_K\}, \tag{16}$$

the least common multiple of $G_1, G_2, \ldots$, and G_K. Also, let $L_i = \text{lcm}\{p_i, q\}$, $1 \leq i \leq K$.

Again, let $\boldsymbol{Y} = \{y_m\}_{m=-\infty}^{\infty}$ be a periodic sequence of period q with $y_m = q \cdot m$ for all m and $\boldsymbol{Y} + k = \{y_m + k\}_{m=-\infty}^{\infty}$ be a shifted version of $\boldsymbol{Y}$. Let

$$\boldsymbol{X} = \bigcup_{1 \leq i \leq K} \boldsymbol{X}_i', \tag{17}$$

such that x is an element of $\boldsymbol{X}$ if and only if it is an element of $\boldsymbol{X}_i'$ for some i, $1 \leq i \leq K$. Clearly, $\boldsymbol{X}$ is a periodic sequence with period $\text{lcm}\{p_1, p_2, \ldots, p_K\}$. Define

$$d(\boldsymbol{X}_i', \boldsymbol{Y} + k) = \min_{-\infty \leq l,m \leq \infty} |y_l + k - x_{i,m} - O_i|, \tag{18}$$

and

$$d(\boldsymbol{X}, \boldsymbol{Y} + k) = \min_{1 \leq i \leq K} \{d(\boldsymbol{X}_i', \boldsymbol{Y} + k)\}. \tag{19}$$

The optimal value of k is again given by (8). It can be easily shown that $d(\boldsymbol{X}, \boldsymbol{Y} + k)$ is periodic with period GL because $d(\boldsymbol{X}_i', \boldsymbol{Y} + k)$ is periodic with period G_i, $1 \leq i \leq K$. As a result, k^* can be chosen to satisfy $0 \leq k^* \leq GL - 1$. The fact that $G_i | q$, $1 \leq i \leq K$, implies $GL | q$. In other words, GL is a factor of q and, therefore, is upper bounded by q. If the new TS is considered as the $(K+1)$th TS when the $(K+2)$th TS is to be scheduled, then we have

$$O_{K+1} = k^* \leq q - 1 = p_{K+1} - 1. \tag{20}$$

This proves the property that O_i satisfies $0 \leq O_i \leq p_i - 1$.

To compute $d(\boldsymbol{X}, \boldsymbol{Y} + k)$, we need $d(\boldsymbol{X}_i', \boldsymbol{Y} + k)$ for all i, $1 \leq i \leq K$. To determine k^*, a scheduling matrix of size $K \times GL$ is constructed, as shown in Fig. 4. The ith row of the scheduling matrix is $[d(\boldsymbol{X}_i', \boldsymbol{Y} + 0) d(\boldsymbol{X}_i', \boldsymbol{Y} + 1) \ldots d(\boldsymbol{X}_i', \boldsymbol{Y} + G_i - 1)]$ repeated for GL/G_i times. Given the scheduling matrix, $d(\boldsymbol{X}, \boldsymbol{Y} + k)$ can be obtained as the minimum element of the kth column. Finally, the optimal value of k is given by the index of the column with the maximum $d(\boldsymbol{X}, \boldsymbol{Y} + k)$.

As derived previously, the complexity of computing $d(\boldsymbol{X}_i', \boldsymbol{Y} + k)$, $0 \leq k \leq G_i - 1$, requires $4 \times \min\{p_i, q\}$ comparisons. The overall complexity to generate the scheduling matrix for K existing TSs is, therefore, $\sum_{i=1}^{K} 4 \times \min\{p_i, q\}$, which is upper bounded by $4 \times q \times K$ comparisons. To find k^*, we need $K \times GL$ comparisons to obtain $d(\boldsymbol{X}, \boldsymbol{Y} + k)$, $0 \leq k \leq GL - 1$, and GL comparisons to determine k^*.

Note that our algorithm is unable to break a tie based on average distance. In case there are multiple choices for k^*, we select the one with the maximum column sum. As a result, it requires $2(K-1)$ additions and one comparison for each tie-breaking. If there is still a tie, it is broken arbitrarily.

Example 1. Consider an example for $K = 2$, $p_1 = 12$, $p_2 = 15$, and $q = 18$. Assume that the TS with period p_1 was scheduled earlier than the TS with period p_2. Since $\boldsymbol{X}_1$ is used as reference, we assign $\boldsymbol{X}_1' = \boldsymbol{X}_1 = \{12 \cdot m\}_{m=-\infty}^{\infty}$. O_2, the offset of $\boldsymbol{X}_2'$ with respect to $\boldsymbol{X}_1'$ has to be determined based on the scheduling algorithm. Since $\gcd(12, 15) = 3$, there are only three possible values for O_2, as shown in Fig. 5. After some calculations, we get $O_2 = 1$ or 2. Assume that we choose $O_2 = 2$. To schedule the third TS with period $q = 18$, we need to compute $d(\boldsymbol{X}_1', \boldsymbol{Y} + k)$, $0 \leq k \leq 5$, and $d(\boldsymbol{X}_2', \boldsymbol{Y} + k)$, $0 \leq k \leq 2$, because $G_1 = \gcd(12, 18) = 6$ and $G_2 = \gcd(15, 18) = 3$. The results are $[d(\boldsymbol{X}_1', \boldsymbol{Y} + 0)\ d(\boldsymbol{X}_1', \boldsymbol{Y} + 1) \ldots d(\boldsymbol{X}_1', \boldsymbol{Y} + 5)] = [0\ 1\ 2\ 3\ 2\ 1]$ and $[d(\boldsymbol{X}_2', \boldsymbol{Y} + 0)\ d(\boldsymbol{X}_2', \boldsymbol{Y} + 1)\ d(\boldsymbol{X}_2', \boldsymbol{Y} + 2)] = [1\ 1\ 0]$.

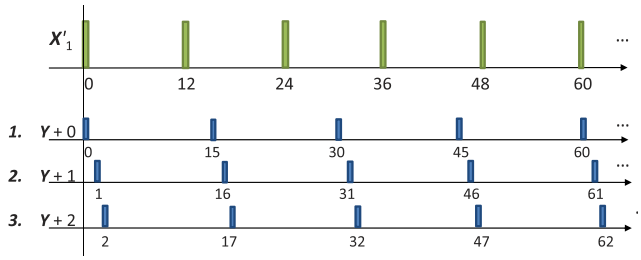

Fig. 5. The three choices for O_2 in Example 1.

Since $GL = \text{lcm}\{6, 3\} = 6$, we have six choices for O_3. Fig. 6 illustrates the relative positions of $\boldsymbol{X}'_1$, $\boldsymbol{X}'_2$, and $\boldsymbol{Y} + k$, $0 \le k \le 5$. For each choice, we need to compare and select the minimum between $d(\boldsymbol{X}'_1, \boldsymbol{Y} + k)$ and $d(\boldsymbol{X}'_2, \boldsymbol{Y} + k)$. Based on our algorithm, the scheduling matrix is of size 2×6 and is given by

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Note that the second row is $[1\ 1\ 0]$ repeated for two times. Given the scheduling matrix, we have $[d(\boldsymbol{X}, \boldsymbol{Y}+0)\ d(\boldsymbol{X}, \boldsymbol{Y}+1) \ldots d(\boldsymbol{X}, \boldsymbol{Y}+5)] = [0\ 1\ 0\ 1\ 1\ 0]$. As a result, the value of k^* can be selected as 1, 3, or 4. The column sums are 2, 4, and 3 for columns 1, 3, and 4, respectively. Therefore, k^* is selected as 3.

4 Handling of Multiple Traffic Classes

4.1 Class-Based Scheduling

In real applications, it is likely that there are only a few possible periods to schedule TSs. Therefore, one can partition TSs into classes such that two TSs are in the same class if and only if they have identical periods of schedule. Assume that there are C classes, called Class 1, Class 2, ..., and Class C. Let p_i represent the period of Class i and n_i the number of TSs in Class i. If $n_i = 0$, then Class i is considered not exist. For ease of description, we assume that $n_i > 0$ for all i, $1 \le i \le C$.

Consider Class i and let $e_1, e_2, \ldots,$ and e_{n_i} be the TSs in the class. A TS, say, e_1, is selected as the representative of Class i. Let $\boldsymbol{X}_i = \{x_{i,m}\}_{m=-\infty}^{\infty}$ be a sequence of period p_i such that $x_{i,m} = p_i \cdot m$ for all m. We shall use the representative TS as reference within the class and, therefore, assign $\boldsymbol{X}_i$ as the scheduled instants of TS e_1. Let $o_{i,s}$ be the intraclass offset of TS e_s with respect to TS e_1. As a result, the scheduled instants for TS e_s is $\boldsymbol{X}_i + o_{i,s}$. Let

$$\boldsymbol{X}_{i,s} = \boldsymbol{X}_i + o_{i,s}, \tag{21}$$

and

$$\bar{\boldsymbol{X}}_i = \bigcup_{1 \le s \le n_i} \boldsymbol{X}_{i,s}. \tag{22}$$

Assume that a new TS of Class j is to be scheduled. The impact of $\bar{\boldsymbol{X}}_i$ to the new TS can be analyzed as follows.

Let $\boldsymbol{Y} = \{y_m\}_{m=-\infty}^{\infty}$ be a periodic sequence of period p_j with $y_m = p_j \times m$ for all m. According to the results presented in the previous section, we need to construct a scheduling matrix of size $n_i \times G_{i,j}$, where $G_{i,j} = \gcd(p_i, p_j)$.

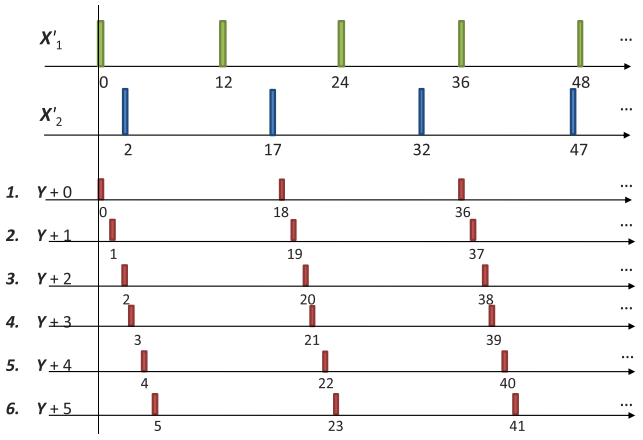

Fig. 6. The six choices for O_3 in Example 1.

The sth row of the scheduling matrix is $[d(\boldsymbol{X}_i, \boldsymbol{Y}+0)\ d(\boldsymbol{X}_i, \boldsymbol{Y}+1) \ldots d(\boldsymbol{X}_i, \boldsymbol{Y}+G_{i,j}-1)]$ circularly shifted to the right by $o_{i,s}$ positions. By taking the minimum element in each column, we obtain

$$R(\bar{\boldsymbol{X}}_i, \boldsymbol{Y}) = [d(\bar{\boldsymbol{X}}_i, \boldsymbol{Y}+0)\ d(\bar{\boldsymbol{X}}_i, \boldsymbol{Y}+1) \ldots d(\bar{\boldsymbol{X}}_i, \boldsymbol{Y}+G_{i,j}-1)]. \tag{23}$$

Note that the optimal SST of the new TS cannot be determined solely by $R(\bar{\boldsymbol{X}}_i, \boldsymbol{Y})$ because there are still TSs in other classes.

Assume that $R(\bar{\boldsymbol{X}}_i, \boldsymbol{Y})$, $1 \le i \le C$, are obtained. We shall use the representative TS of Class 1 as reference of the overall system. Let

$$\bar{\boldsymbol{X}} = \bigcup_{1 \le i \le C} \bar{\boldsymbol{X}}_i. \tag{24}$$

Further, let O_i, $1 \le i \le C$, be the interclass offset of the Class i representative TS with respect to the Class 1 representative TS. To determine the optimal SST of the new TS, we construct a global scheduling matrix M of size $C \times GL_j$, where

$$GL_j = \text{lcm}\{G_{1,j}, G_{2,j}, \ldots, G_{C,j}\}. \tag{25}$$

The ith row of M is $R(\bar{\boldsymbol{X}}_i, \boldsymbol{Y})$ repeated for $GL_j/G_{i,j}$ times and then circularly shifted to the right by O_i positions. By taking the minimum element of each column, we obtain

$$R(\bar{\boldsymbol{X}}, \boldsymbol{Y}) = [d(\bar{\boldsymbol{X}}, \boldsymbol{Y}+0)\ d(\bar{\boldsymbol{X}}, \boldsymbol{Y}+1) \ldots d(\bar{\boldsymbol{X}}, \boldsymbol{Y}+GL_j-1)]. \tag{26}$$

Finally, the optimal scheduled instants of the new TS is given by $\boldsymbol{Y} + k^*$, where k^* satisfies

$$k^* = \arg \max_{0 \le k \le GL_j - 1} \{d(\bar{\boldsymbol{X}}, \boldsymbol{Y} + k)\}. \tag{27}$$

If the new TS is considered as the $(n_j + 1)$th TS of Class j, then we update the intraclass offset

$$o_{j,n_j+1} = k^* - O_j. \tag{28}$$

4.2 Suggested Implementation Method

To reduce online scheduling complexity, we allocate C pairs of arrays for each class. Again, consider Class i. Denote the kth element of the jth pair of arrays by $A_{i,j}[k]$

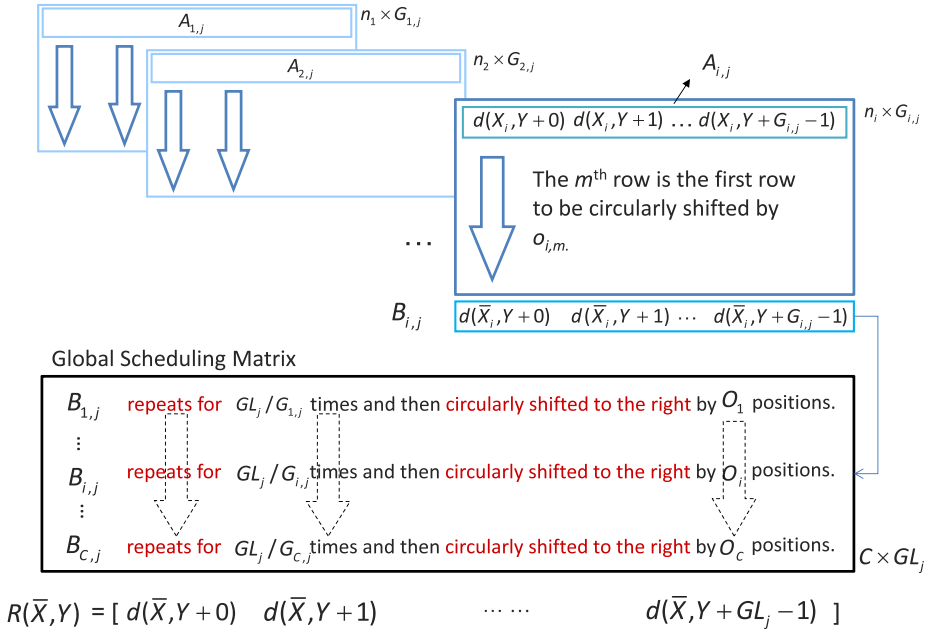

Fig. 7. The illustration of constructing a global scheduling matrix.

and $B_{i,j}[k]$, $0 \le k \le G_{i,j}-1$. The array $A_{i,j}$ stores $[d(\boldsymbol{X}_i, \boldsymbol{Y}+0)\ d(\boldsymbol{X}_i, \boldsymbol{Y}+1) \dots d(\boldsymbol{X}_i, \boldsymbol{Y}+G_{i,j}-1)]$ for $\boldsymbol{Y} = \{y_m\}_{m=-\infty}^{\infty}$ with $y_m = p_j \times m$ for all m. Note that $A_{i,j}$ represents the impact of the representative TS e_1 to a new TS of Class j. The array $B_{i,j}$ stores $R(\bar{\boldsymbol{X}}_i, \boldsymbol{Y})$ and represents the impact of all TSs in Class i to a new TS of Class j. When a new TS of Class j is to be scheduled, the arrays $B_{i,j}$, $1 \le i \le C$, are used to construct the global scheduling matrix M as illustrated in Fig. 7. All we need to do is taking the minimum of each column and then find the maximum among the minima. The complexity is only $C \times GL_j$ comparisons. After the new TS is scheduled, we need to update $B_{j,l}[k]$, $1 \le l \le C$, and $0 \le k \le G_{j,l}-1$, as

$$B_{j,l}[k] = \min\{B_{j,l}[k], A_{j,l}[k - o_{j,n_j+1}]\}, \tag{29}$$

because the impact of TSs in Class j to a new TS of every class is changed. Here, the index $k - o_{j,n_j+1}$ is performed modulo $G_{j,l}$. The complexity of the update process is $\sum_{l=1}^{C} G_{j,l}$ comparisons, which is upper bounded by $C \times p_j$ comparisons.

When a TS in Class i finishes, we need to update $B_{i,j}[k]$, $1 \le j \le C$ and $0 \le k \le G_{i,j}-1$, as follows: remove the finished TS so that the updated n_i, denoted by n_i', becomes $n_i - 1$. Construct a scheduling matrix of size $n_i' \times G_{i,j}$ such that the mth row is $A_{i,j}$ circularly shifted to the right by $o_{i,m}$ positions. The content of $B_{i,j}[k]$ is updated as the minimum of the kth column. Of course, a new representative TS is selected before the update process if the finished one was originally the representative TS of Class i. The complexity of the update process is $n_i' \times \sum_{j=1}^{C} G_{i,j}$ comparisons.

Again, we break a tie based on column sum of the global scheduling matrix which requires $2(C-1)$ additions and one comparison. Note that one can precompute $A_{i,j}[k]$, $1 \le i,\ j \le C$, and $0 \le k \le G_{i,j}-1$. The initial content of $B_{i,j}[k]$ is set to a sufficiently large value for all j and k if there is no Class i TS, i.e., $n_i = 0$.

Example 2. Assume that $C = 2$, $n_1 = 2$, $n_2 = 1$, $p_1 = 6$, and $p_2 = 9$, and a new TS of Class 2 is to be scheduled. Based on the assumptions, we have $G_{1,1} = 6$, $G_{1,2} = 3$, $G_{2,1} = 3$, and $G_{2,2} = 9$. Besides, the contents of $A_{i,j}[k]$ are given by $A_{1,1} = [0\ 1\ 2\ 3\ 2\ 1]$, $A_{1,2} = [0\ 1\ 1]$ and $A_{2,1} = [0\ 1\ 1]$, $A_{2,2} = [0\ 1\ 2\ 3\ 4\ 4\ 3\ 2\ 1]$. Let e_1 and e_2 represent the TSs in Class 1 with e_1 being the representative. Also, let f_1 be the representative TS in Class 2. Assume that TS e_1 was scheduled first, followed by TS f_1, and then TS e_2. As a result, when the new TS f_2 is to be scheduled, we have $O_2 = 1$ (which is randomly selected from 1 and 2) and $o_{1,2} = 3$ (which is randomly selected among 2, 3, and 5). At this moment, the contents of $B_{i,j}[k]$ are given by $B_{1,1} = [0\ 1\ 1\ 0\ 1\ 1]$, $B_{1,2} = [0\ 1\ 1]$; and $B_{2,1} = [0\ 1\ 1]$, $B_{2,2} = [0\ 1\ 2\ 3\ 4\ 4\ 3\ 2\ 1]$. The global scheduling matrix

$$M = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 & 4 & 4 & 3 & 2 \end{bmatrix}.$$

Therefore, the value of k^* can be chosen as 2, 4, 5, 7, or 8. We select $k^* = 5$ because it has maximum column sum. Then we compute $o_{2,2} = k^* - O_2 = 4$ and update $B_{2,1} = [0\ 0\ 1]$, $B_{2,2} = [0\ 1\ 2\ 1\ 0\ 1\ 2\ 2\ 1]$.

5 Performance Evaluation

The considered scenario for our simulations is composed of periodic beacons and five classes of traffic. The five classes of traffic in the system are bidirectional real-time voice, real-time video, streaming audio, streaming video, and gaming. The traffic characteristics, listed in Table 1, are obtained from [4], [6], and [8]. The video traces are available online [13]. It is assumed that there are K STAs, each is configured with a scheduled TS belonging to one of the five classes. The number of STAs is increased in multiples of five STAs to maintain the same number of TSs in each traffic class. The system parameters conform to the Orthogonal Frequency Division

TABLE 1
Traffic Characteristics

Class	Application	Mean Data Rate	Service Interval
1	Real-time Gaming	20 Kbps	100 ms
2	Real-time Voice (G.711 with silence suppression)	26 Kbps	40 ms
3	Real-time Video (MPEG 4 Trace: Lecture Room Cam)	58 Kbps	60 ms
4	Streaming Audio	128 Kbps	150 ms
5	Streaming Video (MPEG 4 Trace: First Contact)	330 Kbps	300 ms

Multiplexing (OFDM) PHY specification [16] and the calculation for frame transmission time can be found in [9]. The repetition period L equals $\mathrm{lcm}\{100, 40, 60, 150, 300\} = 600$ (ms) when all traffic classes, including the beacons, exist in the system. The chosen maximum precision in our simulations is $1\mu s$.

5.1 Comparison of Computational Complexity

For the OAS-APSD algorithm, it needs to insert the already scheduled events within L into the $List_{SE}$ and sort the elements. The complexity of sorting is $\log_2 K \times \sum_{i=1}^{K} L/p_i$ comparisons if the Merge Sort [7] is used. To find the two closest scheduled events for all the scheduled instants given a candidate of SST, by the idea of the Insertion Sort [7], requires $\sum_{i=1}^{K} L/p_i$ comparisons. Since there are q candidates, the overall complexity is $q \times \sum_{i=1}^{K} L/p_i$ comparisons. Computation of relative distances for all the q candidates takes $q \times 2 \times L/q = 2L$ subtractions. To find the minimum relative distances for all the q candidates requires $q \times 2 \times L/q = 2L$ comparisons. Finally, it takes q comparisons to determine the optimal SST. Note that the tie-breaking can be realized by using the sum of relative distances, rather than the average distance. It takes $2L - q$ additions to obtain those sums of relative distances and each tie-breaking needs one comparison. Comparisons of online scheduling complexity are listed in Table 2.

TABLE 2
Online Complexity Comparisons

Algorithm	Prepare distances	Find the minimum distance	Find max of min's	Tie-breaking
OAS-APSD	$q \times \sum_{i=1}^{K} L/p_i + 2L$	$q \times 2 \times L/q$	q	$2L - q$
LCS-APSD	$4 \times q \times C$	$K \times GL$	GL	$2(K-1)$
CLCS-APSD	offline	$C \times GL_j$	GL_j	$2(C-1)$

(In Comparisons/Subtractions/Additions).

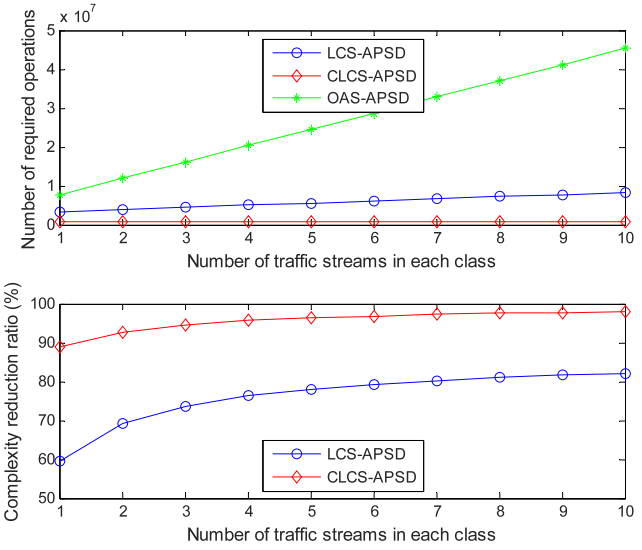

Fig. 8. The performance of complexity reduction.

In addition to complexity analysis, we also provide some numerical results. In the numerical evaluation, we increase the number of existing TS in each class of application, and check the average online complexity of the OAS-APSD algorithm and that of the proposed algorithms. Given the number of existing TSs, the average complexity for finding the SST is derived by averaging the number of necessary online operations when a new TS belonging to each class of application joins. Since the $List_{SE}$ of OAS-APSD could be reused after it is established, the complexity of sorting is ignored here. The Low Complexity S-APSD (LCS-APSD) algorithm refers to the idea described in Section 3.2; however, to reduce complexity, the contents of $d(\boldsymbol{X}_i', \boldsymbol{Y} + k)$'s are reused for the TSs of the same class. Therefore, the complexity in preparing the scheduling matrix for K existing TSs is reduced from $4 \times q \times K$ to $4 \times q \times C$ comparisons. Our proposed algorithm using the suggested implementation method presented in Section 4.2 is referred to as Class-based LCS-APSD (CLCS-APSD) algorithm. The number of required operations and complexity reduction ratios are shown in Fig. 8. Here, the complexity reduction ratios are defined as

$$(N_{OAS\text{-}APSD} - N_{LCS\text{-}APSD})/N_{OAS\text{-}APSD}, \tag{30}$$

and

$$(N_{OAS\text{-}APSD} - N_{CLCS\text{-}APSD})/N_{OAS\text{-}APSD}, \tag{31}$$

where $N_{OAS\text{-}APSD}$ is the number of operations (comparisons and subtractions) required by the OAS-APSD algorithm and $N_{LCS\text{-}APSD}$ and $N_{CLCS\text{-}APSD}$ are those required by LCS-APSD and CLCS-APSD, respectively. As can be seen, the average reduction ratio is as high as about 82 percent for LCS-APSD and 98 percent for CLCS-APSD when there are 50 TSs in the system.

5.2 Comparison of Energy Consumptions

In this evaluation, we fix the number of existing TSs at 50 (10 for each class) and use the OAS-APSD and our proposed algorithms to schedule those TSs. In our simulations, we consider power saving for TSs which require QoS support and HCCA is chosen as the access policy. As a

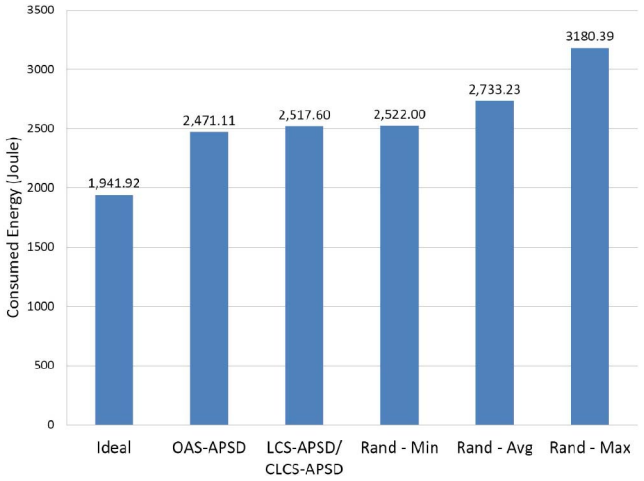

Fig. 9. Comparison of energy consumptions.

consequence, the QAP is responsible for coordinating the channel access and no collision can happen. We assume that the $(5 \times i + j)$th TS belongs to Class j for $0 \leq i \leq 9$ and $1 \leq j \leq 5$. The simulation is performed to model 600 seconds of the real time. The Awake state takes 1.4 W while the Doze state consumes only 0.045 W [14]. The switchover in between the states takes about $250\ \mu s$ [10] and consumes the same power as that in the Awake state. The system parameters conform to the 802.11a and are available in [12]. The PHY data rate is 24 Mbps while the PHY control rate is 6 Mbps. In addition to OAS-APSD and LCS-APSD/CLCS-APSD, we also conduct simulation for the Random algorithm which selects the SST for the newly joined TS randomly and uniformly over $[0,\ SI_{new} - 1]$. An ideal case which assumes no SP overlapping is also presented as a reference.

Comparison of total energy consumption of the 50 STAs for the investigated schemes is provided in Fig. 9. In this figure, Rand-Min, Rand-Avg, and Rand-Max are, respectively, the minimum, average, and maximum energy consumptions for the Random algorithm of 500 simulations. As revealed in Fig. 9, the OAS-APSD performs slightly better than LCS-APSD/CLCS-APSD, which in turn consumes slightly less energy than Rand-Min. The reason OAS-APSD performs slightly better than the proposed LCS-APSD/CLCS-APSD is that it adopts a more complicated tie-breaking scheme based on average distances. However, the difference is not significant. As for the Random algorithm, its performance varies randomly. In our simulations, the proposed LCS-APSD/CLCS-APSD algorithms consume, respectively, about 9 and 26 percent less energy as compared with average and maximum energy consumptions of the Random algorithm. Because of the low online complexity, we believe it is worthwhile to use the proposed CLCS-APSD algorithm for energy saving.

The energy consumption of an STA depends on the time spent for data delivery (including interframe spaces and acknowledgments), the waiting time the STA has to stay awake before transmission, and the number of switchovers during simulations. The waiting time of an STA in a given SP starts from its scheduled wake-up time and covers the duration during which it cannot access the medium because

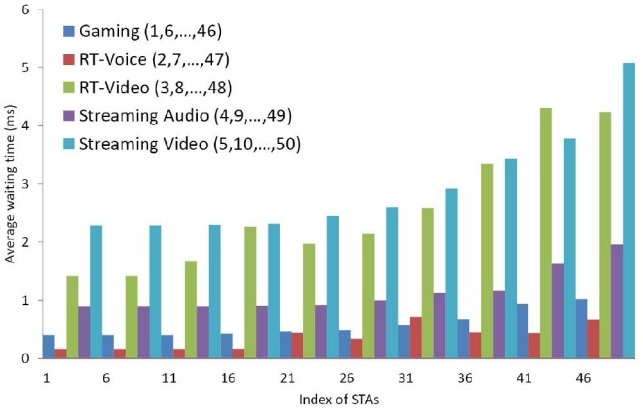

Fig. 10. Average waiting time among STAs.

of the transmissions of previous STAs. In our simulations, we give higher channel access priorities to the TS/STAs which are scheduled earlier, i.e., the STAs with smaller indices. Therefore, the later-order STAs tend to wait longer than the earlier ones. The average waiting time and energy consumption of different STAs for the proposed LCS-APSD/CLCS-APSD are shown in Figs. 10 and 11, respectively.

To explain the results shown in Figs. 10 and 11, the following statistics are helpful. In our simulations, the ideal average SPs for delivering these five classes of applications are 0.5, 0.22, 1.52, 1, and 2.39 ms, while the SIs are 100, 40, 60, 150, and 300 ms, respectively. Fig. 10 shows the waiting time of STAs. In general, the waiting time increases as STA index increases. However, since TSs are added one by one, scheduling of their SSTs may slightly affect the results. According to the results shown in Fig. 11, real-time video consumes the most energy among all classes because it requires a large number of switchovers and long time duration for delivering data. As for streaming video, although it also has long time duration for delivering data, it needs the least number of switchovers among the five classes of applications due to its long SI. Consequently, its energy consumption is moderate. Define the duty cycle of a TS as the ratio of its average SP to its SI. One can easily compute the duty cycles for the five classes of applications as 0.005, 0.006, 0.025, 0.007, and 0.008, respectively. In

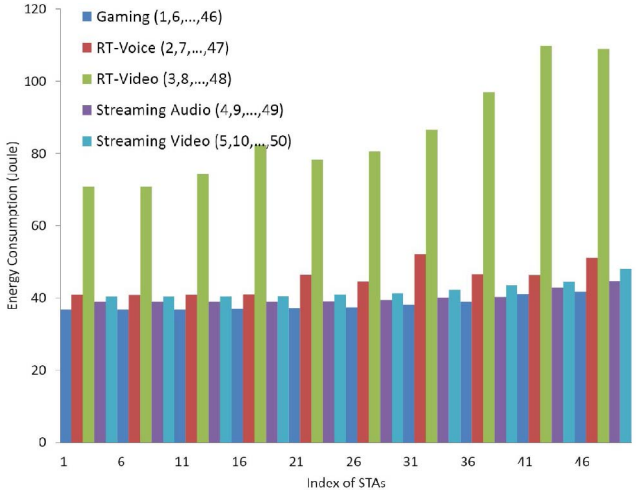

Fig. 11. Energy consumptions among STAs.

general, a larger duty cycle implies more energy consumption. The exception in our simulations is that real-time voice consumes more energy than real-time gaming, streaming audio, and streaming video. The reason is that real-time voice requires a large number of switchovers.

6 Conclusion

Compared with PSM, the S-APSD scheme defined in IEEE 802.11e provides a better mechanism to increase power saving performance when delivering QoS-sensitive traffic. In this paper, we focus on designing a feasible noncontiguous scheduling algorithm to be used for S-APSD. Our design takes advantage of the periodicity property of schedule to largely reduce online computational complexity. We also present an efficient implementation method for class-based systems. As demonstrated in performance comparison, the online computational complexity of our proposed algorithms is much smaller than that of previous related work with comparable energy consumption performance. Some interesting and challenging further research topics such as efficient rearrangement of existing schedule when a new TS is to be added is currently under investigation.

References

[1] *IEEE 802.11 WG: IEEE Standard 802.11-1999, Part 11: Wireless LAN MAC and PHY Layer Specifications, ISO/IEC 8802-11:1999(E),* IEEE, 1999.

[2] *IEEE Std 802.11e-2005, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements,* IEEE, 2005.

[3] 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 Comm. Magazine,* vol. 10, no. 6, pp. 40-50, Dec. 2003.

[4] X. Pérez-Costa, D. Camps-Mur, J. Palau, D. Rebolleda, and S. Akbarzadeh, "Overlapping Aware Scheduled Automatic Power Save Delivery Algorithm," *Proc. European Wireless Conf. (EW),* Apr. 2007.

[5] X. Pérez-Costa, D. Camps-Mur, and T. Sashihara, "Analysis of the Integration of IEEE 802.11e Capabilities in Battery Limited Mobile Devices," *IEEE Wireless Comm. Magazine,* vol. 12, no. 6, pp. 26-32, Dec. 2005.

[6] Q. Zhao and D.H.K. Tsang, "An Equal-Spacing-Based Design for QoS Guarantee in IEEE 802.11e HCCA Wireless Networks," *IEEE Trans. Mobile Computing,* vol. 7, no. 12, pp. 1474-1490, Dec. 2008.

[7] T.H. Cormen, C.R. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms,* second ed. MIT, 2009.

[8] F. Fitzek, A. Koepsel, A. Wolisz, M. Krishnam, and M. Reisslein, "Providing Application-Level QoS in 3G/4G Wireless Systems: A Comprehensive Framework Based on Multi-Rate CDMA," *IEEE Wireless Comm. Magazine,* vol. 9, no. 2, pp. 42-47, Apr. 2002.

[9] Y. Xiao and J. Rosdahl, "Throughput and Delay Limits of IEEE 802.11," *IEEE Comm. Letters,* vol. 6, no. 8, pp. 355-357, Aug. 2002.

[10] J.-R. Hsieh, T.-H. Lee, and Y.-W. Kuo, "Energy-Efficient Multi-Polling Scheme for Wireless LANs," *IEEE Trans. Wireless Comm.* vol. 8, no. 3, pp. 1532-1541, Mar. 2009.

[11] C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multi-Programming in a Hard-Real-Time Environment," *J. ACM,* vol. 20, no. 1, pp. 46-61, 1973.

[12] M.S. Gast, *802.11 Wireless Networks—The Definition Guide.* O'Reilly, 2002.

[13] F.H.P. Fitzek and M. Reisslein, "MPEG-4 and H.263 Video Traces for Network Performance Evaluation," *IEEE Networks,* vol. 15, no. 6, pp. 40-54, http://www-tkn.ee.tu-berlin.de/research/trace/trace.html, Dec. 2001.

[14] E.-S. Jung and N.H. Vaidya, "An Efficient MAC Protocol for Wireless LANs," *Proc. IEEE INFOCOM,* vol. 3, pp. 1756-1764, June 2002.

[15] A. Grilo, M. Macedo, and M. Nunes, "A Scheduling Algorithm for QoS Support in IEEE 802.11e Networks," *IEEE Wireless Comm. Magazine,* vol. 10, no. 3, pp. 36-43, June 2003.

[16] *IEEE Standard for Information Technology - Telecomm. 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,* IEEE, 2007.

[17] IEEE 802.11 Working Group, http://www.ieee802.org/11, 2012.

Tsern-Huei Lee received the BS degree from National Taiwan University, Taipei, ROC, the MS degree from the University of California, Santa Barbara, and the PhD degree from the University of Southern California, Los Angeles, in 1981, 1984, and 1987, respectively, all in electrical engineering. Since 1987, he has been a member of the Faculty of National Chiao Tung University, Hsinchu, Taiwan, where he is a professor in the Department of Communications Engineering and a member of the Center for Telecommunications Research. He serves as a consultant of various research institutes and local companies. His current research interests include network security, broadband switching systems, network traffic management, and wireless communications. He received an outstanding paper award from the Institute of Chinese Engineers in 1991. He is a senior member of the IEEE.

Jing-Rong Hsieh received the BS, MS, and PhD degrees from National Chiao Tung University, Hsinchu, Taiwan, ROC, in 2003, 2005, and 2010, respectively, all in communications engineering. Since November 2010, he has been a senior engineer in protocol standardization at the HTC Corporation in Taipei, Taiwan. His current research interests include power management and quality of service issues of wireless local area networks and wireless cellular networks. He is a member of the IEEE.