Hindawi
Wireless Communications and Mobile Computing
Volume 2017, Article ID 3410350, 12 pages
https://doi.org/10.1155/2017/3410350

WILEY | Hindawi

Research Article

Exploiting Delay-Aware Load Balance for Scalable 802.11 PSM in Crowd Event Environments

Yu Zhang,[1,2] Mingfei Wei,[1] Chen Cheng,[1] Xianjin Xia,[1] Tao Gu,[2] Zhigang Li,[1] and Shining Li[1]

[1]*School of Computer Science, Northwestern Polytechnical University, Xi'an, China*
[2]*School of Computer Science and IT, RMIT University, Melbourne, VIC, Australia*

Correspondence should be addressed to Yu Zhang; zhangyu@nwpu.edu.cn and Zhigang Li; lizhigang@nwpu.edu.cn

Received 19 March 2017; Accepted 1 June 2017; Published 12 July 2017

Academic Editor: Xiaoqiang Ma

This paper presents ScaPSM (i.e., Scalable Power-Saving Mode Scheduler), a design that enables scalable competing background traffic scheduling in crowd event 802.11 deployments with Power-Saving Mode (PSM) radio operation. ScaPSM prevents the packet delay proliferation of previous study, if applied in the crowd events scenario, by introducing a new strategy of *adequate competition* among multiple PSM clients to optimize overall energy saving without degrading packet delay performance. The key novelty behind ScaPSM is that it exploits delay-aware load balance to control judiciously the qualification and the number of competing PSM clients before every beacon frame's transmission, which helps to mitigate congestion at the peak period with increasing the number of PSM clients. With ScaPSM, the average packet delay is bounded and fairness among PSM clients is simultaneously achieved. ScaPSM is incrementally deployable due to only AP-side changes and does not require any modification to the 802.11 protocol or the clients. We theoretically analyze the performance of ScaPSM. Our experimental results show that the proposed design is practical, effective, and featuring with significantly improved scalability for crowd events.

1. Introduction

Energy saving for mobile devices in 802.11 networks has been a crucial issue over the last decade since Wi-Fi communication consumes a significant amount of energy. Although mobile applications have gained increasing popularity in recent years, the capacity of batteries on mobile devices grows at a much slower pace, and the limited battery life has become a bottleneck of enhancing user experience.

The IEEE 802.11 Standard [1] defines a Power-Saving Mode (PSM) for mobile devices to reduce energy consumption for Wi-Fi communication. However, PSM has become inefficient when multiple mobile clients coexist in a network. The competing background traffic among clients introduces significant delays as clients have to wait for others' transmissions, which could generate the extra energy consumption of the waiting clients.

Some recent efforts have been made to address competing background traffic scheduling in a single AP environment [2–5]. These methods optimize contention energy by isolating client traffic into different smaller time slices. However, to avoid large traffic delays, they only divide time slice within the time of one beacon interval, which produces limited number of time slices and thus leads to scalability issues for the strategies, especially when the Wi-Fi network operates in crowd event environments. A salient example is the annual Super Bowl football game in the United States, where approximately 75K attendees descend on a sports stadium for about half a day. During the 2013 Super Bowl game, 700 APs were deployed to provide a significant capacity for handling up to 30,000 simultaneous connections (i.e., with an average of 43 mobile clients accessed to each AP) [6–8].

Unlike a conventional single AP scenario, crowd event environments impose more challenges on traditional competing traffic scheduling, as summarized as follows.

(i) Large-Scale Competition. A large number of clients may simultaneously use a particular AP, and the number of communication channels is relatively limited at a typical AP cell.

(ii) More Fairness Requirement. People tend to use their Wi-Fi devices more than usual during crowd events to either share exciting live information with their friends or access the Internet, needing more fairness than they used to in other environments.

(iii) Widespread User Satisfaction. Poor performance will affect a large number of people and cause widespread user dissatisfaction.

We find that existing efforts for solving competing traffic scheduling focus much on how to save energy by eliminating contention among competing PSM clients, without considering the negative impact on packet delay due to energy conservation. However, in crowd event environments, although it is important to reduce energy cost, it is equally important to ensure users to have good packet delay performance and fairness simultaneously. Hence, we raise an intriguing question: *how to minimize energy consumption while meeting packet delay performance and ensuring fairness on the basis of scalability in crowd events?*

In this paper, we present a novel scheduler, named ScaPSM, as our first attempt to challenge the above problem. We take a fundamentally different approach *rather than reducing or even completely avoiding any contention among competing PSM clients in crowd events*; we seek to smooth the peaks that cause contention. The basic idea behind ScaPSM is to contribute a new strategy of *adequate competition* that exploits delay-aware load balance to control judiciously some competing PSM clients to contend for buffered packets and forces other PSM clients to delay their traffics to mitigate peak period congestion with packet delay deadline aware.

However, it is difficult to find the optimal adequate competition participants in order to meet both energy consumption minimization and performance requirements. We have the two following challenges to solve. First, background applications during crowd events are delay-sensitive (such as gathering group background management processes in screen-off traffic [9]). Hence, delaying client's traffic should not sacrifice the packet delay performance. The second one is that the packets which arrived at an AP usually belong to certain ongoing traffic sessions. Long traffic delays may lead to packet retransmission, which is undesirable. Hence, delaying downlink traffic must not exceed the maximum retry limit.

This paper makes the following contributions.

(i) We identify the scalability issue of competing PSM traffic in crowd event environments and formally model the delay-aware energy optimization problem in 802.11 networks, which is proved NP-hard.

(ii) We propose two algorithms (i.e., ACAA and FPSA) to determine the optimal number of competing PSM clients based on the specific properties of the problem and prove its stability.

(iii) We design a practical online scheduler, named ScaPSM, to minimize energy consumption while meeting both packet delay deadline and fairness among PSM clients in crowd event environments.

(iv) We conduct comprehensive evaluations, and the results demonstrate that, compared to NAPman and 802.11 Standard, ScaPSM achieves good energy saving with both good packet delay performance and fairness. Meanwhile the proposed algorithm achieves very close power saving to that of NAPman with reduction over 20x packet delay and ≤0.5 s traffic delays when the number of PSM clients reach 100. Our algorithm also achieves over 4x better delay fairness compared to 802.11 Standard.

The rest of the paper is organized as follows. In Section 2, we describe the system model and problem formulation. In Section 3, we present the design of ScaPSM. Performance analysis and extensive evaluation are reported in Section 4. Finally, we review the related work in Section 5 and conclude the paper in Section 6.

2. System Model and Problem Formulation

In this section, we first present the system model and then elaborate on how we handle packet delay and energy consumption while ensuring fairness. At last, we formulate the scheduling problem.

2.1. System Model. We consider a competing background traffic scheduling problem in an 802.11 deployment system during crowd events. Our system consists of one AP and m PSM clients ($m \in \mathcal{N}$). We denote $C = \{c_i \mid i \in \mathcal{N}\}$ as a set of PSM clients. For simplicity, we assume that downlink and uplink are separated, and we focus on downlink competing background traffic. In this paper, we first consider *homogeneous clients* that adopt a *static PSM* (SPSM) mechanism. Discussions on the *Adaptive PSM* (A-PSM) mechanism, another popular PSM implementation, will be left for our future studies. In addition, since scheduling the competing background traffic between PSM clients and CAM (i.e., Constant Awake Mode or high power awake mode) clients has been given a solution in [4], in this paper, we turn our attention on the competing background traffic among a large amount of PSM clients.

According to the 802.11 specification, at the beginning of each beacon interval (denoted by b_i, $i \in \mathcal{N}$), the AP notifies the PSM clients of the presence of buffered packets for them, through the Traffic Indication Map's (TIM's) field in the beacon frame. We assume that the packets of each client arrive continually over time.

2.2. Buffered Data Retrieval Model. Consider the data packets retrieval procedure between the AP and its associated clients in a beacon interval. We assume that every PSM client wakes up for beacon frame at the beginning of the beacon interval. For any client, if the corresponding TIM field in the beacon frame is set, it stays in wake mode and prepares to send a PS-Poll request frame by contending for the channel with other clients; otherwise it goes back to a low-power sleep mode to conserve power. If it wins the contention, the PSM client sends out a PS-Poll and the AP responds to it with a buffered data frame. The PSM clients remain in wake mode until the last packet is delivered, and then it goes back to sleep mode

immediately. We denote the beacon interval that a packet p arrives at the AP as $\mathscr{B}_a(p)$, and the beacon interval that p is scheduled to send to corresponding client as $\mathscr{B}_s(p)$.

We define the capacity of a beacon interval as the maximum amount of data that can be transmitted between clients and the AP in the beacon interval. Let $c(b_i)$ denote the capacity of beacon interval b_i and $v_p(b_i)$ the data transfer rate of packet p in beacon interval b_i. We have the following constraint.

$$\sum_{p\in\{p|\mathscr{B}_s(p)=b_i\}} v_p(b_i) \le c(b_i). \quad (1)$$

2.3. Packet Delay Impact. In order to reduce the energy consumption of client contentions in a beacon interval, a competing traffic scheduler can send traffic based on absolute isolation strategy on condition that there requires no change to the existing 802.11 protocol. Since the PSM clients which have not been selected to retrieve downlink data have to sleep and wait until the next beacon interval, they may suffer from long delay. This kind of packet delay may even violate certain performance bounds such as the *deadline* of a packet. To capture its performance impact, we introduce a performance cost metric $\phi_p(\cdot)$ from a packet point of view, which is exploited from [10].

For simplicity, we assume that any downlink packet can tolerate the same level of traffic delay. When the delay expectation for a buffered packet is violated, its performance may degrade significantly. This would cause bad user experience and thus a large performance cost. We take the term *deadline* as the bound of tolerable waiting delays of a packet.

Note that the packet delay is mainly caused by the MAC contention delay during the beacon interval (i.e., this time is referred to as *competing-beacon packet delay*) and the sleep delay of deferring competing for access (i.e., this time is referred to as *sleep-beacon packet delay*); we define the *performance degradation function* ϕ_p as

$$\phi_p(\text{delay}) = f_{\text{sleep}}(\text{delay}) \times f_{\text{comp}}(\text{delay}) \times \text{Len}(p), \quad (2)$$

where $\text{Len}(p)$ denotes the size of packet p. The function f_{sleep} represents the sensitivity of p to the *sleep-beacon packet delay*, and the function f_{comp} represents the sensitivity of p to *competing-beacon packet delay*. Denoting $\mathscr{B}_d(p)$ as the deadline of p, we can easily get the following property.

Property 1. Any $\phi_p(\cdot)$ should satisfy the following conditions:

(i) $\phi_p(0) = 0$.

(ii) If $d_1 < d_2$, then $\phi_p(d_1) \le \phi_p(d_2)$.

(iii) If $d_1 \le \mathscr{B}_d(p) - \mathscr{B}_a(p) < d_2$, then $\phi_p(d_1) < \phi_p(d_2)$.

The first two conditions ensure that ϕ_p captures the nondecreasing feature between the performance cost and packet delay. The third condition reflects the cost associated with the violation of deadline; that is, the user may have significantly worse experience and thus higher performance degradation cost.

Let $\mathbb{P}$ be a set of pending retrieval packets. Given $\phi_p(\cdot)$ for all the packets in $\mathbb{P}$, we can evaluate the total packet delay performance cost $\Phi(\mathbb{P}, S(\mathbb{P}))$ caused by a schedule $S(\mathbb{P})$ as $\sum_{p\in\mathbb{P}} \phi_p(\mathscr{B}_s(p) - \mathscr{B}_a(p))$. The schedule $S(\mathbb{P})$ is formulated by $S(\mathbb{P}) = \mathbb{P} \times \Gamma = \{\langle p, b_i\rangle\}$, $p \in \mathbb{P}$, $b_i \in \Gamma$, where tuple $\langle p, b_i\rangle$ signifies packet p is scheduled at the beacon interval b_i and $\Gamma = \{b_1, b_2, \ldots, b_m\}$ is a set of continuous beacon intervals during which the packets in $\mathbb{P}$ should be scheduled.

2.4. Competing PSM Clients' Fairness. Fairness is a key design objective for a competing traffic scheduler. As mentioned before, fairness should be considered in terms of both energy and delay. For energy fairness among competing PSM clients, it is ensured by the DCF scheme [11] if all clients have the same physical data rate, since the probability for each client to win channel contention is equivalent in the 802.11 Standard. Thus, in this paper, we focus on *delay fairness* whose meaning is that each client should receive fair opportunity to be scheduled by the AP before the corresponding deadline, irrespective of the number of clients.

Before formally providing a delay fairness metric, we first introduce three definitions about delay fairness among PSM clients as follows.

Definition 2 (delay of packet). Define the time of a buffered packet $d_p = b_s(p) - b_a(p)$ to denote its residential time or delay time at the AP.

Definition 3 (delay of client). The delay time D_i of a client c_i is the longest delay time among all buffered packets of c_i. Let p_{ij}^b and n_i^b represent client c_i's jth packet buffered in the AP and the number of buffered packets of c_i at the bth beacon interval, respectively; then $D_i = \max_{j=1}^{n_i^b} d_p(ij)$.

Definition 4 (delay fairness among competing clients). Given a time period Γ and the set of competing clients, one calls this kind of relation among these competing clients delay fairness if the delays of all competing clients are equal; that is, $D_i = D_j$, $\forall i, j \in \mathcal{N}$ and $i \ne j$.

It is worth noting that the definition of delay fairness above is strict. In the future, we can relax this strict definition and allow different clients having different delay tolerances based on specific application traffic and user's preference. For simplicity in this paper, we intensively consider the delay fairness model defined above.

To measure how well a competing traffic scheduler satisfies the delay fairness, we use the following *Relative Delay Fairness Bound* as a delay fairness metric based on [12].

Definition 5 (relative delay fairness bound). Let $C(\Gamma)$ be the set of clients that are delayed in a given time period Γ. Let ω_i be the weight of client c_i. We use RDFB to stand for Relative Delay Fairness Bound, which is defined as

$$\text{RDFB} = \sup_{i,j\in C(\Gamma)} \left| \frac{D_i(\Gamma)}{\omega_i} - \frac{D_j(\Gamma)}{\omega_j} \right|. \quad (3)$$

RDFB bounds the gap of delays experienced by any two clients in any given time period. Intuitively, the smaller the

gap is, the fairer the scheduler achieves. One of our objectives is to design a competing traffic scheduler with small RDFB.

2.5. Competing Traffic Energy Consumption. An 802.11 radio with PSM operation typically has three basic states: ACTIVE (i.e., TX/RX), IDLE, and SLEEP. We denote the corresponding radio power as P_A (or $P_{\text{tx}}/P_{\text{rx}}$), P_I, and P_S, respectively. As aforementioned, a PSM client wakes up for beacon frame at the beginning of a beacon interval. Each client decides to enter into one of the three states based on the TIM bit settings and results of channel contentions. Specifically, if a client's TIM field is set and it wins the contention, it will enter the high power ACTIVE state to download its data packets by means of PS-Poll → DATA → ACK frame sequences. While if a client's TIM field is set but it fails the contention, it will enter an IDLE state until it succeeds in accessing the wireless channel. Hence, the power consumption in IDLE state is much higher than that in SLEEP state and slightly lower than that in ACTIVE state. We assume that the energy consumption in SLEEP state is negligible since the radio is powered off. For simplicity, we neglect the energy consumption of changing client's radio states.

The competing traffic energy consumption is composed of the energy consumption in ACTIVE state and that in IDLE state. We first estimate the energy consumption in ACTIVE state. For any given size of data units, the data transmission energy depends on the product of two factors: the transmission power and the time taken to transmit all the data bits. Let $t_r(p)$ denote the average time allocated to a client that has one pending packet to retrieve. Based on [13], when there is no PS-Poll collision or transmission corruption, $t_r(p)$ can be expressed as

$$t_r(p) = \frac{CW_{\min}}{2} + T_{\text{pspoll}} + 2\text{SIFS} + T_{\text{data}} + T_{\text{ack}} + \text{DIFS}, \tag{4}$$

where $CW_{\min}$ presents the average back-off time for clients to send a PS-Poll frame, T_{pspoll} represents the time taken by a client to send a PS-Poll frame, T_{data} represents the time taken by the AP to send a DATA frame to the client, T_{ack} represents the time taken by a client to send an ACK frame to the AP, and both SIFS duration and DIFS duration are constants defined by 802.11 protocol.

Let $E_{\text{trans}}(p)$ denote the average packet transmission energy consumed during $t_r(p)$. Then it can be computed as

$$E_{\text{trans}}(p) = t_r(p) P_A. \tag{5}$$

In a given beacon interval b_i, we define three sets C_s^b, C_c^b, and C_f^b to denote the set of clients whose TIM field was set by the AP, the set of clients that successfully retrieved all buffered packets, the set of clients that only retrieved part, instead of all, of the buffered packets, respectively. We use N_c^b to stand for the total number of buffered packets that was successfully received by the clients during beacon interval b_i. Let r_i^b denote the number of client i's buffered packets that still remain at the AP; then we can compute N_c^b as

$$N_c^b = \sum_{i \in C_c^b} n_i^b + \sum_{i \in C_f^b} \left(n_i^b - r_i^b\right). \tag{6}$$

Let $E_{\text{trans}}(b_i)$ denote the total packet transmission energy consumed during a beacon interval b_i; then it can be expressed as

$$E_{\text{trans}}(b_i) = \sum_{p \in \mathbb{P}_c^b} E_{\text{trans}}(p). \tag{7}$$

Thus, during time period Γ, given $\mathbb{P}$ and a schedule $S(\mathbb{P})$, the total packet transmission energy can be estimated as

$$\widetilde{E}_{\text{tran}}(\mathbb{P}, S(\mathbb{P}), \Gamma) = \sum_{p \in \mathbb{P}} E_{\text{trans}}(b_i). \tag{8}$$

We now compute the energy consumption in ILDE state. Let us define three variables $M_s^b \in \mathcal{N}$, $M_c^b \in \mathcal{N}$, and $M_f^b \in \mathcal{N}$ to denote the number of clients in C_s^b, C_c^b, and C_f^b, respectively. According to the Buffered Data Retrieval Model, we can have $M_s^b = M_c^b + M_f^b$. As mentioned before, when only one PSM client wins the contention and is retrieving data frames from the AP, the transmission can be successfully performed. During this transmission time, the rest of clients whose TIM field was set should stay in IDLE state and consume idle power (i.e., P_I). Hence, let $E_{\text{idle}}(p)$ denote the idle energy consumed during an average packet transmission time $t_r(p)$; then it can be computed as

$$E_{\text{idle}}(p) = \left(M_s^b - 1\right) t_r(p) P_I. \tag{9}$$

Note that a background PSM client may quit contentions when it is indicated that no more frames are pending at the AP. It is difficult to determine the number of contending clients in the network on each round of contention. For simplicity, we assume that there are always M_s^b contenders in a given beacon interval b_i.

Let $E_{\text{idle}}(b_i)$ denote the total idle energy consumed by all the contending clients during b_i; then it can be expressed as

$$E_{\text{idle}}(b_i) = \sum_{p \in \mathbb{P}_c^b} E_{\text{idle}}(p). \tag{10}$$

Thus, under the schedule of $S(\mathbb{P})$, the total idle energy during the transmission of P can be estimated as

$$\widetilde{E}_{\text{idle}}(\mathbb{P}, S(\mathbb{P}), \Gamma) = \sum_{p \in \mathbb{P}} E_{\text{idle}}(b_i). \tag{11}$$

2.6. Problem Formulation. Our objective is to find a schedule $S(\mathbb{P})$ that can minimize the total energy consumption for transmitting buffered data in $\mathbb{P}$ without the packet delay performance degradation. It is constrained below an upper bound (denoted by $\widetilde{\Phi}$) during a given time period Γ. That is,

$$\sum_{p \in \mathbb{P}} \phi_p \left(\mathscr{B}_s(p) - \mathscr{B}_a(p)\right) \leq \widetilde{\Phi}. \tag{12}$$

A higher performance bound suggests a longer tolerable delay. Based on (8) and (11), during a given time period Γ, the total energy consumption of all PSM clients can be calculated as $E(\mathbb{P}, S(\mathbb{P}), \Gamma) = \widetilde{E}_{\text{tran}}(\mathbb{P}, S(\mathbb{P}), \Gamma) + \widetilde{E}_{\text{idle}}(\mathbb{P}, S(\mathbb{P}), \Gamma)$.

In order to formulate the optimization problem, we first need to introduce variables $x_{i,j} \in \{0, 1\}$. If $x_{i,j} = 1$, it represents that the i packet in $\mathbb{P}$ (denoted by p_i) is scheduled at the beacon interval b_j; otherwise it is not scheduled.

Then we model the problem as below.

$$\min \quad E(\mathbb{P}, S(\mathbb{P}), \Gamma) \tag{13}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{i,j} = 1, \quad i = 1, 2, \ldots, n \tag{14}$$

$$\sum_{i=1}^{n} x_{i,j} \cdot \text{Len}(p_i) \le c(b_j), \quad j = 1, 2, \ldots, m \tag{15}$$

$$\sum_{i=1}^{n} \left(\sum_{j=1}^{m} j \cdot x_{i,j} - \mathscr{B}_a(p_i) \right) \le \widetilde{\Phi}, \tag{16}$$

where (14) stands for the fact that the ith packet should be scheduled at only one determined beacon interval b_i during the time period of Γ; capacity constraint (15) and delay cost constraint (16) are corresponding to (1) and (12), respectively.

It is worth noting that variables $x_{i,j}$ can only be set to integer one or zero. Hence, the problem modeled in (13)–(16) is an integer programming problem. It is *NP-hard* proved by [14]. Thus, we try to look for an approximate solution, instead of finding the optimal solution.

3. Scheduling Analysis and Algorithms

In this section, we first introduce our scheduling analysis to build a new mathematical model for making clear our target. After that, we design an online scheduler, named ScaPSM (i.e., Scalable Power-Saving Mode Scheduler), aiming to be implemented for real deployment. ScaPSM accounts for the packet delay performance and also ensures fairness among multiple PSM clients as the number of competing clients increases.

3.1. Delay-Aware Energy Scheduling Analysis. The proposed model in (13)–(16) demands that all traffic information in future time window Γ must be available. However, this assumption is limited in practical scenarios since future traffic information of clients cannot be perceived (only historical and present traffic information can be accessed by AP).

We present ScaPSM which does not require any future information of clients. It makes AP track the progress of each client's buffer and adaptively schedules traffic for them. Specifically, ScaPSM makes scheduling decisions in each beacon interval to obtain *adequate competition*, that is, optimizing both the energy and delay performance while ensuring fairness.

Generally, the delay sensitivity of a packet is reflected by its *deadline*. As aforementioned in Section 2.3, large *sleep-beacon packet delay* may even lead to violate the *deadline* of a packet. Therefore, in order to design a delay-aware scheduler, we require any packet to be delivered before a delay upper bound L; that is, $d_p = \mathscr{B}_s(p) - \mathscr{B}_a(p) \le L$.

At any given beacon interval b_i, in order to get *adequate competition* among multiple PSM clients to optimize the overall energy saving without degrading packet delay performance, we need to choose the clients with minimum total energy consumption for data delivering and leave others to stay in sleep state. Therefore, *the key problem we are facing is: given any beacon interval b, how to determine the set* C_s^b *defined in Section 2.5*?

Note that the PSM clients that have not been selected to be in wake mode should wait for the next beacon interval to transmit their PS-Poll requests. This implies that if C_s^b has a small number of clients, the average sleep-beacon packet delay significantly increases. On the other hand, if C_s^b has a large number of clients, these clients may consume significantly power owing to severe contention among themselves.

Therefore, we must judiciously control the qualification and number of competing PSM clients before every beacon frame's transmission. Specifically, in order to determine the set C_s^b, on the one hand, we need to avoid excessive clients to be scheduled at the same beacon interval; on the other hand, the packet delay of clients d_p should not be larger than L. This is essentially a load balance problem at the time range from b_i to b_{i+L-1}. We model the load balance problem as a *min–max of the number of participant problems*.

Before formal formulation, we first introduce the definition of client's remaining time.

Definition 6 (remaining time of client). The remaining time $DL(i)$ of a client c_i is the difference between the packet delay upper bound and the delay of client c_i. Given the packet delay upper bound (L) and the delay of client (D_i), then $DL(i) = L - D_i$.

To take into account the delay performance in our scheduling, according to the remaining time of a client, we first classify the PSM clients into L groups $\mathbb{G} = \{G_1, G_2, \ldots, G_L\}$, with each group $G_j = \{c_i \mid c_i \in \{DL(i) = j - 1\}\}$ $(1 \le j \le L)$. Let $M_j = |G_j|$ represent the number of clients in group G_j. In order to mitigate the congestion at the competing background traffic peak, some clients in G_i may be shifted to G_j $(i > j)$ at earlier beacon intervals for being scheduled. We use a variable M_{ij} to stand for the number of the shifted clients. The variable M_{ij} should meet the scheduling deadline constraint which is expressed by

$$M_{ij} = 0, \quad \text{if } (i \le j). \tag{17}$$

We define k_i to denote the number of clients being arranged to be scheduled at beacon interval of group G_i. k_i can be computed by

$$k_i = M_i + \sum_{j=1}^{L} M_{ji} - \sum_{l=1}^{L} M_{il}, \tag{18}$$

where $\sum_{j=1}^{L} M_{ji}$ stands for the number of clients which has been shifted into group G_i and $\sum_{l=1}^{L} M_{il}$ represents the number of clients which has been shifted out of group G_i.

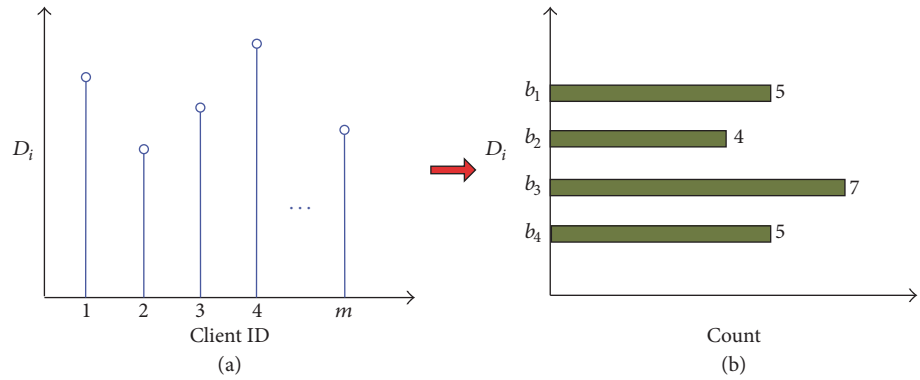

FIGURE 1: Organize clients into L groups.

Therefore, we model the min–max of the number of participant problems as follows.

$$\begin{aligned} \min \quad & \max_{i=1}^{L} k_i \\ \text{subject to} \quad & \text{Constraints (17) and (18).} \end{aligned} \tag{19}$$

Note that the scheduling of PSM client's traffic delivery is controlled by setting TIM bit in a beacon frame. We perform the competing background traffic scheduling for PSM clients at each beacon interval by two steps. In the first step, we determine the number of competing clients (denoted by k_i). In the second step, we select k_i right clients to be scheduled, with the goal of delay fairness.

3.2. Adequate Competition Assignment Algorithm. We start by computing k_i through a load balancing water-filling framework. The basic idea is described as follows.

Consider the L beacon intervals (denoted by $b_1, \ldots, b_L$) starting from the current beacon interval b_i. We use $\mathcal{U}_t$ to denote an *adequate competition set sequence*. Formally, $\mathcal{U}_t = \langle k_1, \ldots, k_L \rangle$, where k_i denotes the number of clients being arranged to be scheduled at b_i. As shown in Figure 1, we organize the clients into L groups, with each group $G_j = \{c_i \mid c_i \in \{DL(i) = j - 1\}\}$ $(1 \le j \le L)$. We initially arrange the clients in G_j to be scheduled at b_j (i.e., set $k_j = M_j$). We consider k_i as the water level of beacon interval b_i and attempt to shift clients from high water-level beacon intervals to low-level ones until water levels of the L beacon intervals can finally reach a *stable state*, just like water flowing. Specifically, restricted by traffic deadlines, we only allow water in b_i flowing forwardly to b_j's $(j < i)$. In this way, if too many clients are arranged to be scheduled at b_i, we will reschedule some of them to be in wake mode at earlier beacon interval for load balancing. Before proceeding, we formally define the *stable state* in L beacon interval as follows.

Definition 7 (stable state). Given an adequate competition scheduling arrangement $\mathcal{U}_t = \langle k_1, \ldots, k_L \rangle$, one says the L beacon intervals are in a stable state if $k_i \ge k_j$ is satisfied for any two beacon intervals b_i and b_j $(1 \le i < j \le L)$. In this case, one calls $\mathcal{U}_t$ a *stable adequate competition scheduling*.

Let $\mathbb{U}$ be the complete set of stable adequate competition scheduling arrangement. We have the following lemma and corollary.

Lemma 8. *Given $\mathcal{U}_t^* = \langle k_1^*, \ldots, k_L^* \rangle$, if $k_1^* = \max_{i=1}^{L}\{k_i^*\}$ and $k_1^* \le k_1$ is satisfied for any $\mathcal{U}_t \in \mathbb{U}$, $\mathcal{U}_t = \langle k_1, \ldots, k_L \rangle$, then $\mathcal{U}_t^*$ is the optimal adequate competition scheduling arrangement.*

Proof. We prove this lemma by using contradiction. We assume $\mathcal{U}_t^*$ is not optimal and denote the real optimal arrangement by $\mathcal{U}_t'$. $\mathcal{U}_t'$ does not satisfy $(k_1' = \max_{i=1}^{L}\{k_i'\} \wedge k_1' \le k_1)$ for any $\mathcal{U}_t \in \mathbb{U}$. There exist two cases.

Case A. If $k_1' \ne \max_{i=1}^{L}\{k_i'\}$, suppose $k_j' = \max_{i=1}^{L}\{k_i'\}$. We have $k_j' > k_{j-1}'$. In this case, by shifting $(k_j' - k_{j-1}')/2$ clients from b_j to b_{j-1}, we can produce another arrangement $\mathcal{U}_t''$ with $k_j'' = k_{j-1}'' = (k_j' + k_{j-1}')/2$. Since $k_j'' < k_j'$, $\mathcal{U}_t''$ is a better arrangement. This contradicts the assumption that $\mathcal{U}_t'$ is optimal.

Case B. If $(k_1' = \max_{i=1}^{L}\{k_i'\} \wedge k_1' > k_1)$ for some $\mathcal{U}_t \in \mathbb{U}$, since $k_1 = \max_{i=1}^{L}\{k_i\}$ under $\mathcal{U}_t$, $\mathcal{U}_t$ is a better arrangement than $\mathcal{U}_t'$. This, again, yields the contradiction.

This completes the proof. □

Corollary 9. *Given $\mathcal{U}_t^* = \langle k_1^*, \ldots, k_L^* \rangle$, if $\mathcal{U}_t^* \in \mathbb{U}$ and $k_1^* \le k_1$ is satisfied for any $\mathcal{U}_t \in \mathbb{U}$, then $\mathcal{U}_t^*$ is optimal.*

Corollary 9 suggests finding optimal arrangements within $\mathbb{U}$. In what follows, we will elaborate our algorithm to compute the optimal adequate competition scheduling arrangement.

The algorithm starts from the initial arrangement of $\mathcal{U}_t^1 = \langle M_1, \ldots, M_L \rangle$. It incrementally performs *stabilizing operations* from b_1 to b_L as follows: based on $\mathcal{U}_t^1$, we attempt to change water levels in b_1 and b_2 (i.e., k_1 and k_2) into a stable state, with minimal increase in k_1. And this produces $\mathcal{U}_t^2$. Next, based on $\mathcal{U}_t^2$, we further stabilize water levels among b_1, b_2, and b_3. Proceeding as above, it will finally cover b_L and produce a stable arrangement $(\mathcal{U}_t^L)$ with minimum k_1.

We describe one step of the operations in detail. Generally, we will produce $\mathcal{U}_t^i$ from $\mathcal{U}_t^{i-1}$. Note that beacon intervals

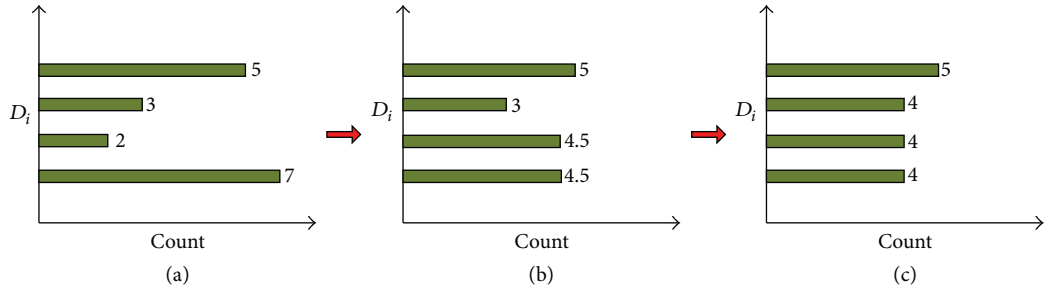

Figure 2: Illustration of stabilizing operations.

$b_1, \ldots, b_{i-1}$ are already in a stable state under $\mathcal{U}_t^{i-1}$. We want to push such stable state further to beacon interval b_i. In case that $k_i > k_{i-1}$ ($k_i = M_i$ under $\mathcal{U}_t^{i-1}$), we need to shift "water" from b_i to earlier beacon intervals for load balancing. This is achieved by the *stabilizing operations*, which require detailed discussions below.

We conduct stabilizing operations to determine the amount of water (M_{ij}) flowing from b_i to each b_j ($1 \le j < i$). We compare k_i with k_{i-1}. If $k_i \le k_{i-1}$, beacon intervals from b_1 to b_i are already in a stable state, and we do not need any operations. If $k_i > k_{i-1}$, we first try to balance water levels in b_{i-1} and b_i. We let $(k_i - k_{i-1})/2$ units of water flow from b_i to b_{i-1} (i.e., $M_{i,i-1} = (k_i - k_{i-1})/2$) and get $k_i' = k_{i-1}' = (k_i + k_{i-1})/2$. Next, we go back to check if the increase of k_{i-1} will break the stable state in $b_1, \ldots, b_{i-1}$. If $k_{i-1}' \le k_{i-2}$, the above operations successfully produce a stable state in beacon intervals from b_1 to b_i. Otherwise, we reattempt to balance water levels in the beacon intervals ranging from b_{i-2} to b_i. In this case, we will produce $k_{i-2}' = k_{i-1}' = k_i' = \overline{K}_3$, where $\overline{K}_3 = \sum_{j=i-2}^{i} k_j/3$. And hence, $M_{lb} = \overline{K}_3 - k_j$, $i-2 \le j < i$. We continue the process until the water levels in $b_1, \ldots, b_i$ become stable.

We illustrate the above operations by an example. As shown in Figure 2, $\mathcal{U}_t^3 = \langle 5, 3, 2, 7 \rangle$. We compute $\mathcal{U}_t^4$ from $\mathcal{U}_t^3$. Since $k_3 < k_4$, we balance water levels in b_3 and b_4 and get $k_3' = k_4' = 4.5$ (see Figure 2(b)). As $k_2 < k_3'$, we further balance water levels in b_2, b_3, and b_4, which produces $k_2' = k_3' = k_4' = 4$ (see Figure 2(c)). In this case, as $k_1 > k_2'$, the 4 beacon intervals reach a stable state. We obtain $\mathcal{U}_t^4 = \langle 5, 4, 4, 4 \rangle$. The amount of water flows from b_4 to b_3 and b_2 are $M_{4,3} = 2$ and $M_{4,2} = 1$, respectively.

Now, we give the algorithm in Algorithm 1. It is clear that it needs to be executed L^2 loops. We note that L is a constant given in advance. Hence, the computation complexity of the algorithm is $O(0)$. As the algorithm ensures k_1 to be increased minimally during each stabilizing operation, according to Corollary 9, the computed adequate competition scheduling arrangement is optimal.

3.3. Fair Participant Selection Algorithm. Based on the results produced by Algorithm 1, we move on to address the scheduling problem by selecting the right k competing clients to be scheduled. The main idea of Algorithm 2 is that scheduler can fairly select the right competing clients by computing dynamic priority weight ω_i for each client c_i. Note that, in Algorithm 1, k_i's and M_{ij}'s may not be integers. Hence, we derive k as $k = \lceil k_1 \rceil$. We select the k clients as follows: (i) all clients in G_1 must be scheduled; (ii) for clients in G_i ($i > 1$), we select the $\lfloor M_{i1} \rfloor$ clients with highest weight $\omega_i = n_i^b/DL(i)$ to ensure delay fairness scheduling defined in Section 2.4. Our strategy is described in Algorithm 2.

Input: M_i $(1 \le i \le L)$.
Output: $\mathcal{U}_t = \langle k_1, \ldots, k_L \rangle$, and M_{ij} $(1 \le i \le L,\ 1 \le j < i)$.
(1) $k_i \leftarrow M_i$, $(i = 1, \ldots, L)$.
(2) **for** $i = 2$ to L **do**
(3) $\quad k_i' \leftarrow k_i$.
(4) $\quad$ **for** $j = i - 1$ to 1 **do**
(5) $\quad\quad$ **if** $k_j \ge k_{j+1}'$ **then**
(6) $\quad\quad\quad$ let $k_l \leftarrow k_l'$, $(l = j+1, \ldots, i)$, and beak the loop.
(7) $\quad\quad$ **end if**
(8) $\quad\quad$ $k_l' \leftarrow \sum_{l=j}^{i} k_l/(i - j + 1)$, $(l = j, \ldots, i)$.
(9) $\quad\quad$ $M_{il} \leftarrow k_l' - k_l$, $(l = j, \ldots, i-1)$.
(10) $\quad$ **end for**
(11) **end for**

Algorithm 1: Adequate competition assignment algorithm (ACAA).

Input: k_1, M_{i1} $(1 < i \le L)$.
Output: $\mathcal{W}_t$ the set of clients scheduled at beacon interval b.
(1) $k \leftarrow \lceil k_1 \rceil$.
(2) count $\leftarrow |G_1| + \sum_{i=2}^{L} \lfloor M_{i1} \rfloor$.
(3) Compute $\omega_i = n_i^b/DL(i)$ for each $c_i \in \mathcal{G}$.
(4) Add all clients in G_1 to $\mathcal{W}_b$.
(5) **for** $i = 2$ to L **do**
(6) $\quad$ Add the $\lfloor M_{i1} \rfloor$ clients with highest ω_i in G to $\mathcal{W}_b$.
(7) **end for**
(8) Select (k – count) clients with highest ω_i from the all clients in $\mathcal{G}$, and add them to $\mathcal{W}_b$.

Algorithm 2: Fair participant selection algorithm (FPSA).

4. Performance Evaluations

In this section, we evaluate the performance of ScaPSM through stability analysis and simulations.

4.1. Stability Analysis. To demonstrate the stability of ScaPSM, we need to show two proofs. First, we should show that there exists one (equilibrium) state at which, once hit, the system will stay forever. Second, we should show that the system will move to the equilibrium state eventually regardless of its initial or current state.

Theorem 10. *The equilibrium state is delay-aware overall energy-optimal, and it achieves min–max of the number of participants at any beacon interval.*

Clearly, in the equilibrium state, the scheduler in the AP completes an adequate competition scheduling arrangement procedure by using Algorithm 1. Then, the statement is true according to Lemma 8.

4.2. Methodology and Simulation Setup. We compare ScaPSM with both 802.11 Standard [1] and NAPman [4]. Using an absolute isolation strategy, NAPman can create no contention wireless channel access for PSM clients. Thus, it may be regarded as an optimal energy saving scheduler for competing background traffic.

Three metrics are used for performance evaluation, that is, *energy consumption*, *packet delay,* and *delay fairness*. Through our analysis, ScaPSM can achieve all three desirable properties. Specifically, when PSM clients increase, ScaPSM should optimize overall energy consumption without degrading packet delay while ensuring fairness among PSM clients.

First, we start with controlled PSM client traffic in order to highlight various aspects of ScaPSM. This controlled traffic is intended to represent the worst case for various scheduling strategies since there is a packet in every beacon interval for a PSM client. Second, to compare ScaPSM with NAPman and 802.11 Standard in crowd events, we configure the server to send packets with a random interval ranging from 10 ms to 300 ms (based upon the SIGCOMM'08 traces [15], primarily Web traffic). In a typical crowd event scenario, according to the report in [6], each AP is associated with 107 clients on average, where up to 43 clients may simultaneously access Wi-Fi. To simulate such highly-competitive environment, we employ one AP and up to 100 clients in our OMNet++ simulation. We configure parameters with $T_b = 100$ ms, $L = 500$ ms, where T_b is the duration of one beacon interval. Each data point is an average over 20 independent runs.

4.3. Impact of Number of PSM Clients on Power Consumption. We first investigate the impact of the number of PSM clients on power consumption of one single client and all clients. The results of controlled traffic and trace-driven traffic are shown in Figures 3(a)-3(b) and Figures 3(c)-3(d), respectively.

In Figures 3(a)-3(b), we can see that as the number of PSM clients increases, both ScaPSM and 802.11 Standard consume higher power than NAPman (i.e., an approach with no contention) in the cases of both one single client and all clients. However, the power consumption of ScaPSM increases much slower than that of 802.11 Standard. This is easy to understand because the total number of competing clients of 802.11 Standard is equivalent to the total number of PSM clients if all clients have buffered packets in the AP. Since the adequate competition strategy of ScaPSM can control the number of competing clients at any beacon interval, the power consumption caused by contention reduces correspondingly. Moreover, the power consumption of ScaPSM is very close to that of NAPman. In Figures 3(c)-3(d), we see that, under the trace-driven traffic, with the number of PSM clients increasing, the power drawn by one single client and all clients with ScaPSM is still much lower than that of 802.11 Standard. The results are similar to that of the controlled traffic configuration.

4.4. Impact of Number of PSM Clients on Packet Delay. We design experiments to study the impact of the number of PSM clients on packet delay performance of our algorithms. The results of controlled traffic and trace-driven traffic are shown in Figures 4(a)-4(b) and Figures 4(c)-4(d), respectively.

Figure 4(a) shows that the packet delay of ScaPSM is almost unchanged as the number of PSM clients increases. The adequate competition assignment algorithm of ScaPSM achieves good performance. In contrast, NAPman performs the worst. The average packet delay of no contention becomes very high as the number of PSM clients increases. When the number of PSM clients is 40, the average packet delay of no contention is around seven times as large as that of ScaPSM. Due to the large difference between no contention and our algorithm, the logarithmic function is used to have a fine-grained view, as shown in Figure 4(b). The result shows that the absolute isolation strategy of NAPman optimizes energy consumption at the cost of large packet delays. This is because in order to eliminate power consumption caused by contentions, NAPman will generate a large amount of *sleep-beacon packet delay*. Since ScaPSM is delay-aware, the average packet delay is bounded by deadline L. As shown in Figure 4(b), the packet delay of 802.11 Standard is the lowest when the number of PSM clients increases. This is because ScaPSM has to delay some clients' traffic by the right duration to mitigate traffic congestion. As a result, ScaPSM produces small *sleep-beacon packet delays*. Figures 4(c)-4(d) show that, under trace-driven traffic, ScaPSM and NAPman exhibit similar performance. This shows that ScaPSM achieves good scalability.

4.5. Impact of Number of PSM Clients on Delay Fairness. Finally, we confirm experimentally that ScaPSM provides good delay fairness, irrespective of the number of PSM clients. The results of controlled traffic and trace-driven traffic are shown in Figures 5(a) and 5(b), respectively. As the absolute isolation strategy of NAPman can be considered as a round-robin scheme, we exclude NAPman from the delay fairness comparison.

From Figure 5(a), we can see that ScaPSM has a better delay fairness performance than 802.11 Standard, no matter under controlled traffic or trace-driven traffic. The RDFB value of ScaPSM remains almost a small constant. In contrast, the RDFB value of 802.11 Standard increases 4x larger than that of ScaPSM when the maximum number of clients changes from 50 to 100. This is because ScaPSM is delay-aware, and the difference between the weights ω_i and ω_j of any two clients is constrained. In contrast, 802.11 Standard

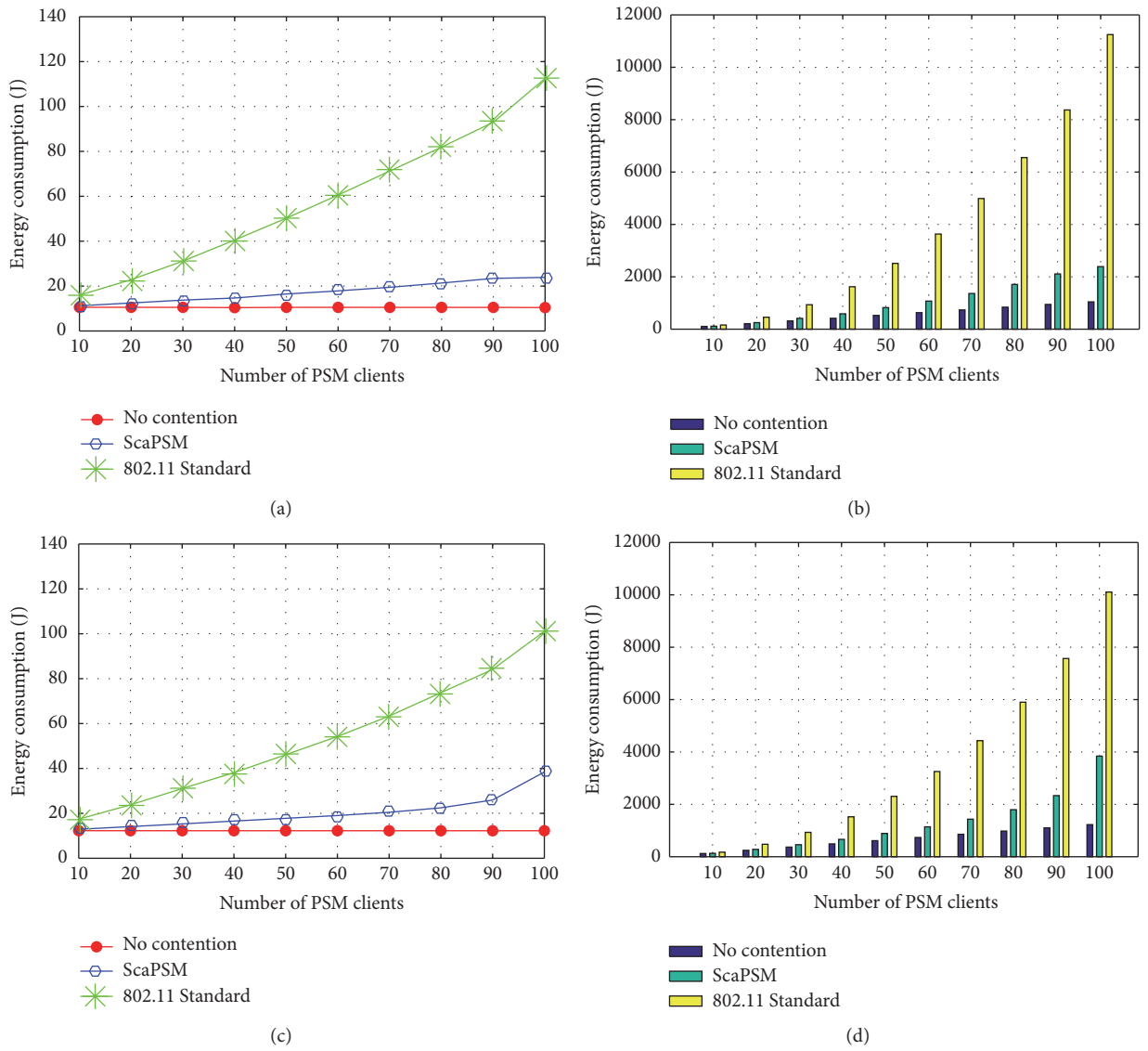

FIGURE 3: (a) Power drawn by a static PSM versus number of PSM clients under controlled traffic; (b) overall power drawn by all static PSMs versus number of PSM clients under controlled traffic; (c) power drawn by a static PSM versus number of PSM clients under trace-driven traffic; (d) overall power drawn by all static PSMs versus number of PSM clients under trace-driven traffic.

has no concept of delay deadline; therefore, its difference between any two clients' weights is large. Similar results are also observed under trace-driven traffic, as shown in Figure 5(b).

5. Related Work

5.1. Crowd Event Scenario. Network communication in crowd events has recently attracted much research attentions. Shafiq et al. [16] and Erman and Ramakrishnan [7] take the first step to study traffic characteristics in the crowd event scenario. Although they do not propose strategies for performance improvement, their work provides crucial insights into the design of ScaPSM. There are a few techniques, such as WiFox [8] and AMuSe [17], being proposed to improve system throughput in dense AP/client environments. However, they do not address the energy issues of 802.11 network. Our work essentially fills the gap.

5.2. Contention Avoidance Scheduling. This issue has been extensively studied in [2–5] to save energy. Time slicing is used in [2] to make each PSM client's packets delivered only in its appointed time slice to save power and reduce the effect of background traffic. In LAWS [3], the AP advertises a subset of PSM clients in the beacon and clients use information in the beacons to determine their polling sequence to avoid client contention. However, these solutions require modifications to the 802.11 Standard, and thus, changes to both mobile clients and APs are unavoidable. SOFA [5] maximizes the total sleep time of all clients. However, SOFA assumes that AP has full control of its downlink traffic which is often limited in the 802.11 Standard since the AP shares the

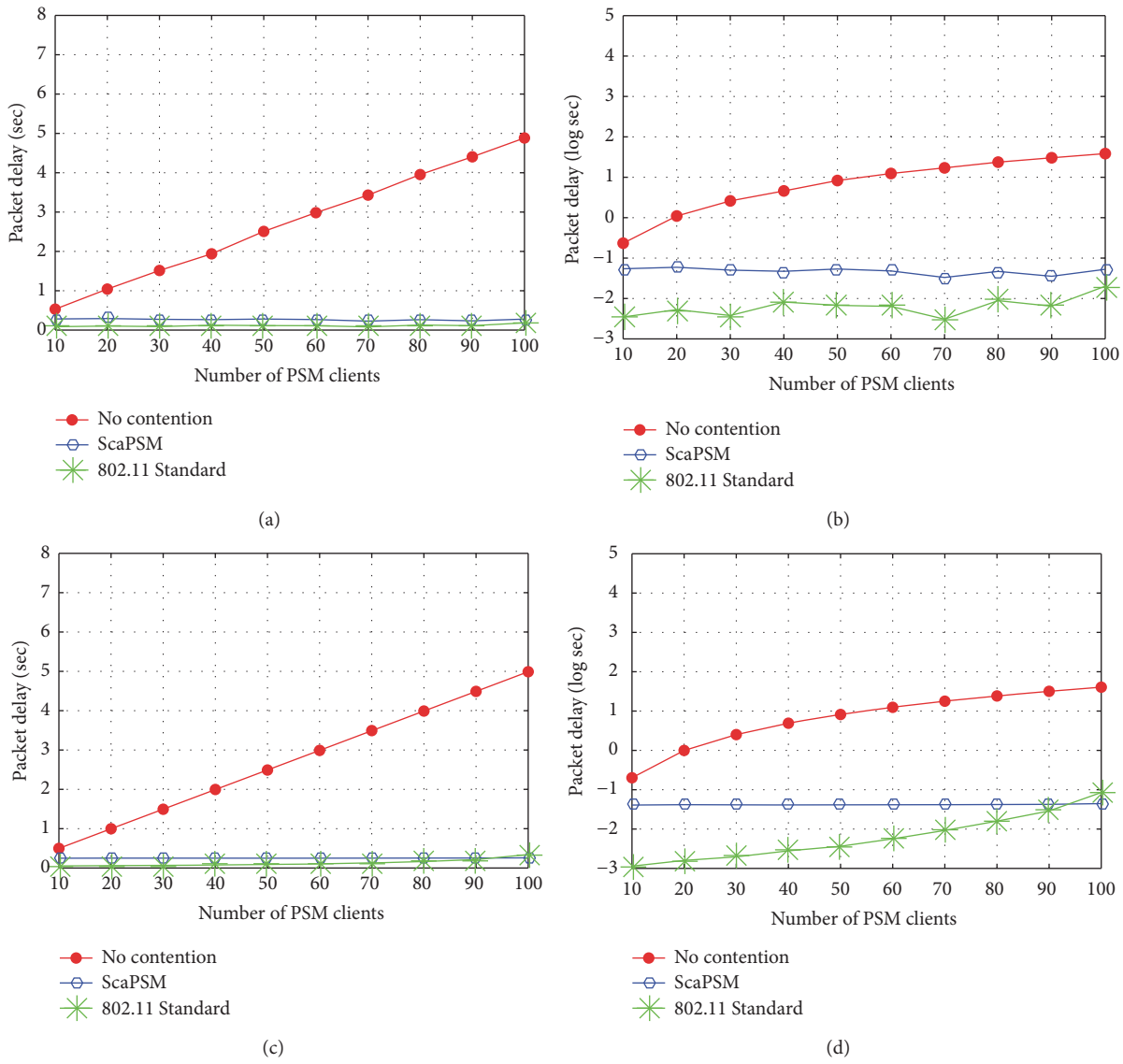

FIGURE 4: (a) Packet delay versus number of PSM clients under controlled traffic; (b) packet delay versus number of PSM clients by logarithmic function under controlled traffic; (c) packet delay versus number of PSM clients under trace-driven traffic; (d) packet delay versus number of PSM clients by logarithmic function under trace-driven traffic.

channel access with associated clients equally. NAPman [4] implements a new energy-aware fair scheduling algorithm at AP to minimize Wi-Fi radio wake-up time and eliminate unnecessary retransmissions in the presence of competing traffic. Further, NAPman leverages AP virtualization to make different PSM clients wake up at staggered time intervals, so that these clients can monopolize wireless channel and receive TIM separately. However, previous efforts generally consider how to save energy by reducing or even eliminating contention among competing PSM clients and also ignore the negative impact on packet delay performance due to energy conservation. Moreover, all these solutions are not for crowd events scenario where scalability is considered as a major issue. SleepWell [18] coordinates the activity circles of multiple APs to allow clients to sleep longer, and therefore this technique may be complementary to ScaPSM.

5.3. Beacon Management Method for WLAN. Lee et al. [13] propose a beacon management scheme that restricts the number of nodes in wake mode in each beacon interval with the maximum number of packets to be delivered according to the transmission duration. However, they only consider the power efficiency when network congestion occurs, instead of the whole views, such as the trade-off between energy consumption and delay performance. EDP [19] provides an analytical model for energy consumption and packet delay in highly congested 802.11 networks and proposes a power-saving strategy to determine the number of PSM clients in wake mode that balances energy consumption and packet delay. However, EDP does not consider fairness issues among PSM clients. Moreover, these schemes are not for crowd event scenarios where a large amount of PSM clients with the delay-aware requirement simultaneously compete for access.

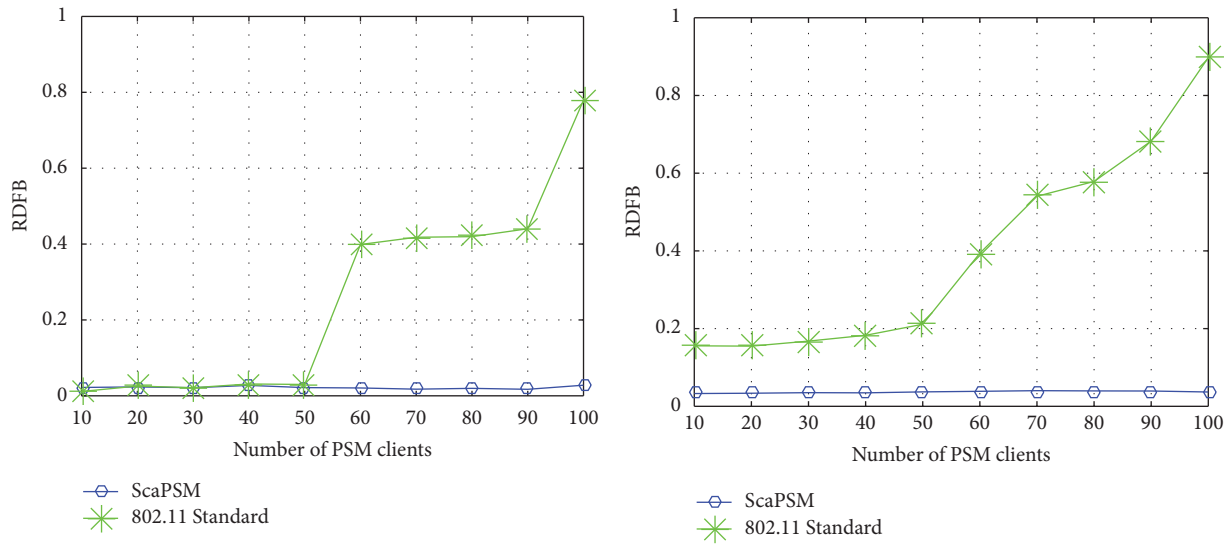

FIGURE 5: (a) Delay fairness versus number of PSM clients under controlled traffic; (b) delay fairness versus number of PSM clients under trace-driven traffic.

6. Conclusions

In this paper, we address the energy issues of mobile devices in 802.11 networks for crowd events. We propose an online competing background traffic scheduling algorithm to improve the client energy efficiency, while ensuring the packet delay performance. Different from existing work, we formulate the delay performance degradation problem and build a comprehensive metric to capture the impact of delay performance and delay fairness. Our evaluation results demonstrate the effectiveness of our proposed schemes in achieving better performance over existing work. We further validate the high energy saving of our proposed scheduling algorithm with increasing the number of PSM clients through controlled and trace-driven simulations. In our future work, we will investigate the impact of application traffic and heterogeneous mobile devices for crowd events.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] LAN/MAN Standards Committee, S. Committee, and I. Computer, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, vol. 2012, 2012.

[2] Y. He and R. Yuan, "A novel scheduled power saving mechanism for 802.11 wireless LANs," *IEEE Transactions on Mobile Computing*, vol. 8, no. 10, pp. 1368–1383, 2009.

[3] H.-P. Lin, S.-C. Huang, and R.-H. Jan, "A power-saving scheduling for infrastructure-mode 802.11 wireless LANs," *Computer Communications*, vol. 29, no. 17, pp. 3483–3492, 2006.

[4] E. Rozner, V. Navda, R. Ramjee, and S. Rayanchu, "NAPman: network-assisted power management for WiFi devices," in *Proceedings of the 8th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '10)*, pp. 91–105, San Francisco, Calif, USA, June 2010.

[5] Z. Zeng, Y. Gao, and P. R. Kumar, "SOFA: A sleep-optimal fair-attention scheduler for the power-saving mode of WLANs," in *Proceedings of the 31st International Conference on Distributed Computing Systems, ICDCS 2011*, pp. 87–98, July 2011.

[6] Super bowl plans to handle 30,000 wi-fi users at once and sniff out 'rogue devices'. http://arstechnica.com/information-technology/2013/02/super-bowl-plans-to-handle-30000-wi-fi-users-at-once-and-sniff-out-rogue-devices/, 2013.

[7] J. Erman and K. K. Ramakrishnan, "Understanding the super-sized traffic of the super bowl," in *Proceedings of the ACM IMC'13*, pp. 353–359, 2013.

[8] A. Gupta, J. Min, and I. Rhee, "WiFox: scaling WiFi performance for large audience environments," in *Proceedings of the 8th ACM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '12)*, pp. 217–228, ACM, December 2012.

[9] J. Huang, F. Qian, Z. M. Mao, S. Sen, and O. Spatscheck, "Screen-off traffic characterization and optimization in 3G/4G networks," in *Proceedings of the 2012 ACM Internet Measurement Conference, IMC 2012*, pp. 357–363, November 2012.

[10] Y. Cui, S. Xiao, X. Wang, M. Li, H. Wang, and Z. Lai, "Performance-aware energy optimization on mobile devices in cellular network," in *Proceedings of the 33rd IEEE Conference on Computer Communications, IEEE INFOCOM 2014*, pp. 1123–1131, May 2014.

[11] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.

[12] W. Wang, B. Liang, and B. Li, "Low complexity multi-resource fair queueing with bounded delay," in *Proceedings of the 33rd IEEE Conference on Computer Communications, IEEE INFOCOM 2014*, pp. 1914–1922, May 2014.

[13] J. R. Lee, S. W. Kwon, and D. H. Cho, "A new beacon management method in case of congestion in wireless lans," in *Proceedings of the IEEE VTC'05*, pp. 12–15, 2005.

[14] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*, pp. 85–103, 1972.

[15] A. Schulman, D. Levin, and N. Spring, 2008, Crawdad data set umd/sigcomm2008.

[16] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang, "A first look at cellular network performance during crowded events," in *Proceedings of the 2013 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS 2013*, pp. 17–28, June 2013.

[17] Y. Bejerano, J. Ferragut, K. Guo et al. et al., "Scalable WiFi multicast services for very large groups," in *Proceedings of the ICNP'13*, pp. 1–12, 2013.

[18] J. Manweiler and R. R. Choudhury, "Avoiding the rush hours: WiFi energy management via traffic isolation," in *Proceedings of the ACM MobiSys'11*, pp. 253–266, 2011.

[19] D. Jung, R. Kim, and H. Lim, "Power-saving strategy for balancing energy and delay performance in WLANs," *Computer Communications*, vol. 50, pp. 3–9, 2014.