State Transition Reduction for Sleep Scheduling in IEEE 802.16e Mobile Subscriber Stations

Li-Hsing Yen, Chih-Yuan Cheng, Yulin Su

Abstract—IEEE 802.16e devices may enter sleep mode to conserve energy. There has been some work studying how to schedule device's packet transmissions to maximize the device's sleep period while meeting associated QoS (quality of service) requirements. This paper proposes two schemes that attempt to save more energy for a given sleep schedule by accelerating or deferring scheduled packet transmissions with the intention to further reduce the frequency of transitions from sleep to active modes (states). Performance evaluation results indicate that the proposed schemes effectively reduce state transitions when compared with existing sleep scheduling schemes. Packet delays may also be improved in case of accelerating packet transmissions.

Keywords—WiMAX; sleep scheduling; QoS; state transitions

1 Introduction

IEEE 802.16e [1] or WiMAX is a standard for wireless Metropolitan Area Networks which provide broadband wireless access services to roaming users. Besides high-speed data transmissions, the current version of IEEE 802.16e also supports handoffs of user equipments, security enhancement, and power-saving techniques.

WiMAX networks provide connection-oriented data service. In a WiMAX network, a mobile subscriber station (MSS) may simultaneously establish several connections with the serving base station (BS). Each connection can have its own QoS (quality of service) demand. Energy saving is a critical issue for battery-powered MSSs. To conserve energy, an MSS may turn off its transceiver (i.e., entering sleep mode) when communication services are not needed. IEEE 802.16e defines three power-saving classes (Types I, II, and III) that provide different sleep mode activation patterns to meet different QoS demands. Type I power-saving class alternates sleep and listening intervals. In listening intervals, an MSS is made aware of whether any packet addressed to it is queued at the BS. If such packet exists, the MSS recovers from sleep mode to retrieve the packet. Otherwise, the MSS enters sleep mode again with a doubled sleep interval. This design aims at minimizing energy consumption on idle listening for non-realtime applications. Type II class also alternates sleep and listening intervals but it allows data exchange during the listening interval without leaving sleep mode. The ratio of listening interval to the whole is fixed to accommodate periodic packets from real-time applications. Type III class lets MSS sleep for a predefined period and then enter back to active mode. It suits best for idle MSSs for which only periodic ranging signaling is required.

The ratio of sleep period to the whole, referred to as *sleep ratio*, has been used to measure the efficiency of power saving. Achievable sleep ratio in a WiMAX network depends on packet generation/arrival patterns and power-saving class in use. When an MSS enters sleep mode, all packets destined for the MSS will be buffered in the BS serving the MSS. These packets will be sent to the MSS after the MSS goes backs to active mode. Therefore, the activation of sleep mode saves energy on one hand while increases packet delivery latency on the other hand. The length of the latency is in general proportional to the length of the sleep period.

In the literature, some studies have considered sleep scheduling for multiple MSSs [2], [3], [4], which demands that the active period of each MSS must be exclusive. Some studies have considered sleep scheduling for one or more CBR (Constant Bit Rate) connections within an MSS. Jang et al. [5] investigated the application of Type I power-saving class to a single CBR connection. Other researchers considered applying Type II power-saving class to an MSS with multiple CBR connections [6], [7]. Their attempts are feasible only when the length of sleep interval is less than the minimal tolerable packet delays of all connections [7]. If connections have diverse characteristics, Type II power-saving class may gives rise to low sleep ratio. To improve, sleep and listening intervals should be changed dynamically. Aperiodic On-Off Scheme (AS) [6] and Minimum Wakeup Time (MWT) [2] are two protocols proposed for this problem.

Both AS and MWT do not consider energy spent on transitions between sleep and active modes. When a transceiver switches back from sleep mode to working mode, it needs some time (called startup time) to scale up its internal clock rates and optionally perform frequency calibration. Thus a mode (state) transition takes time and also extra energy. As we shall show in this paper, reducing the number of state transitions could also save energy on applying Type II power-saving class to realize a sleep schedule, or reduce the number of active instances of Type II power-saving class.

The purpose of this work is to reduce state transitions by merging two or more separated sleep periods into one. We consider multiple CBR connections within an

MSS that may have diverse traffic patterns and QoS requirements. The proposed schemes take sleep schedules yielded by AS or MWT, and attempt to further reduce state transitions while respecting packet order and packet delay constraints. We conducted extended simulations to investigate the effectiveness of the proposed schemes. The results indicate that the proposed schemes achieve the design goal when compared with existing sleep scheduling schemes. The impact of the proposed schemes on packet delay was also studied.

The remainder of this paper is organized as follows. The next section reviews related work and points out the limitation of existing approaches. Sec. 3 presents two approaches to state transition reductions. Experimental results are presented in Sec. 4. The last section concludes this paper.

2 PRELIMINARIES

2.1 Background and Related Work

In IEEE 802.16e, both BS and MSS can initiate sleep mode. If an MSS initiates the sleep mode, it should send request messages to BS to coordinate related parameters such as initial sleep period and maximum sleep interval.

Transmission paths between a BS and associated MSS can be uplink (from MSS to BS) or downlink (from BS to MSS). In this work, we assume 802.16e Time Division Duplex (TDD) mode, where data transmissions within a frame are divided into two parts (subframes), one for downlink and the other for uplink. For applications that demand two-way CBR traffic such as voice over IP (VoIP), packet scheduling need consider only one direction of data transmissions (either downlink or uplink). The scheduling of the other part is assumed symmetric.

In the literature, there has been some work on energy efficiency improvement for generic BS-based [8] and IEEE 802.11 [9] wireless networks. For IEEE 802.16e networks, some researches have analyzed the performance of sleep mode operations [10], [11], [12]. Existing sleep scheduling schemes can be classified into two types. The first type of sleep scheduling, referred to as *inter-MSS* sleep scheduling, considers the scheduling among multiple MSSs where only a single downlink or uplink channel is available for packet transmissions. As all MSSs contend for the exclusive use of the communication channel, the problem is to arrange an non-overlapping schedule that minimizes energy consumption. In Ref. [3], an MSS in sleep mode should be periodically awaken to guarantee a minimal data rate that serves as a QoS requirement. On the other hand, an MSS in the awake state should complete its transmission as soon as possible to minimize the time and thus the energy spent on idle waiting. The authors proposed to allocate almost all the channel bandwidth to one MSS (called primary MSS) and just enough bandwidth to other MSSs (called secondary MSSs) to guarantee their minimum data rate requirements. The goal is to minimize the primary MSS's state transitions and the idle waiting time of secondary MSSs. In a follow-up work the authors considered the same setting with additional inclusion of multicast messages in the schedule [4].

The abovementioned work does not explicitly specify which power-saving class is used. MMPS (Multiple MSSs Power-Saving Scheduler) [2] considered non-overlapping sleep scheduling among multiple MSSs using power-saving class Type II. This work differs from [3], [4] in that it takes packet delay constraint rather than minimal data rate as the only QoS parameter under consideration. Moreover, MMPS acquires traffic patterns from the knowledge of inter-packet arrival time. In contrast, the work in [3], [4] learns of traffic conditions from the length of the packet transmission queue. Extensions of MMPS [2] put packets from several MSSs into one frame to increase bandwidth utilization. A limitation of MMPS is the assumption of one CBR connection per MSS.

Inter-MSS sleep scheduling demands that the transmission activation time of each MSS should be exclusive. In fact, an IEEE 802.16e network typically provides several subchannels such that several transmissions toward or from different MSSs can be done simultaneously. The applicability of the inter-MSS sleep scheduling is therefore limited.

The second type of sleep scheduling, termed *intra-MSS* sleep scheduling, focuses on one or more CBR connection residing in an MSS. Jang et al. [5] assumed a CBR connection within an MSS and studied how to set the initial sleep interval for Type I power-saving class to maximize power saving while still meeting packet delay constraint. Their experimental results indicated that packet latency is proportional to I_0 while energy consumption is inversely proportional to I_0, where I_0 is the length of the initial sleep interval. For CBR traffic, the authors suggested that I_0 should be less than inter-packet arrival time (plus allowable delay jitter) to meet packet delay constraint. The limitation of this study is that only one MSS and one CBR connection were taken into account. The case of multiple connections within an MSS was not considered. Also, using Type I power-saving class to deal with CBR traffic is not appropriate.

When two or more CBR connections coexist in an MSS, the MSS should be awake whenever a connection calls for the transceiver. Consequently, establishments of multiple CBR connections may give rise to poor sleep ratio. Sleep ratio can be improved if transmission time of packets can be scheduled without breaking possible QoS constraints. Consider an MSS with four CBR connections, where packet arrival patterns of these connections are shown in Fig. 1(a). If no special treatment is done (packets are transmitted at the earliest possible time, which is typically when they arrive at the transceiver), the resultant traffic pattern will be that shown in Fig. 1(b). We can see that the MSS should be awake in 14 out of 20 frames. If each connection can tolerate a packet delay that is at least equal to its inter-packet arrival time, we may schedule the transmission

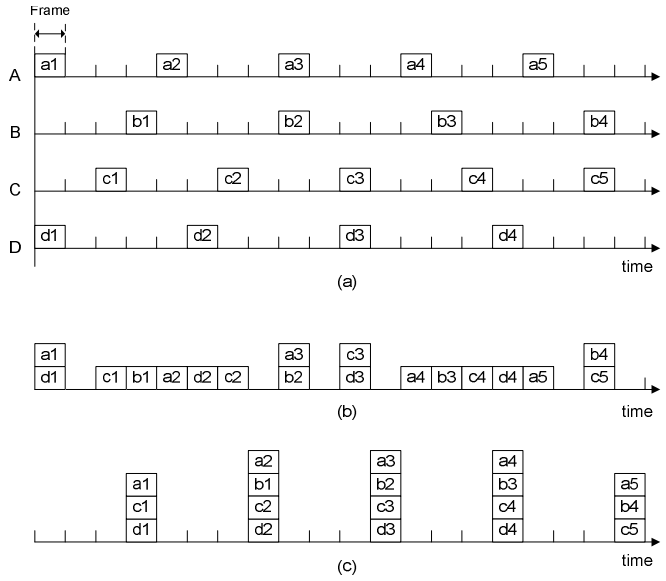

Fig. 1. (a) Exact packet arrival pattern (b) Traffic pattern without scheduling (c) Traffic pattern of some schedule

of packets to yield a higher sleep ratio, as shown in Fig. 1(c). This possibility is also based on the fact that an MSS can be allocated adequate bandwidth to accommodate the transmission of multiple packets in a single frame.

AS [6] and MWT [2] are two approaches that attempt to maximize MSS's sleep ratio by scheduling multiple CBR connections with different QoS requirements and traffic patterns. AS first sorts all connections belonging to the same MSS according to the connection's packet delay constraint, and arranges packet transmission time for each connection based on the sorted order. In determining a packet's transmission schedule, AS takes the nearest possible *active frame* with priority, where an active frame is a frame that has already accommodated at least one packet's transmission. If no active frame can be found, AS chooses the latest possible frame within the delay constraint with the intention to maximize the likelihood of the scheduled packet being transmitted together with other packets in the same frame.

Unlike AS, MWT aims at increasing frame utilization while minimizing packet delivery latency. All packets, upon their arrivals, will be put in a transmission buffer first. Buffered packets are transmitted under two conditions. The first is when these packets together can run out of the bandwidth capacity of a frame that is allocated to the MSS. The second is right before any of these packets violates its delay constraint. Consequently, packets will be put together in the earliest possible frame such that transmissions of these packets together can run out of the frame's capacity yet respect delay constraint. If this is impossible, packets will be transmitted together in the latest possible frame such that the delay constraint of any packet is not violated.

TABLE 1
QoS Parameters of four CBR connections coexisting in an MSS

Connection ID	Inter-packet arrival time (frame)	QoS delay constraint (frame)
A	1	3
B	1	3
C	2	3
D	2	4

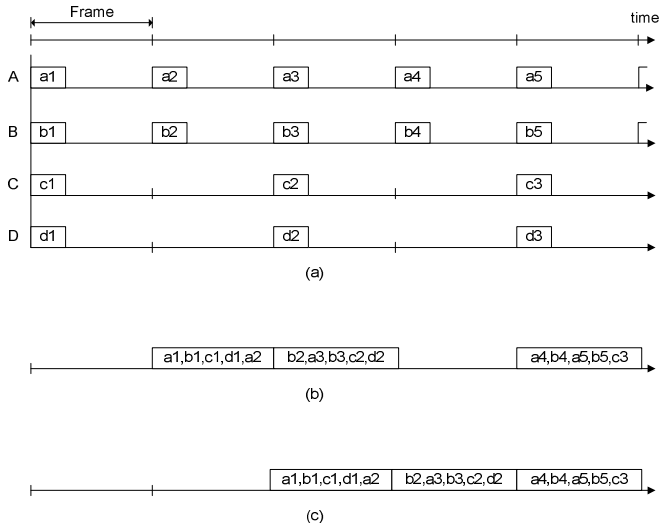

Fig. 2. (a) Packet arrival pattern (b) Schedule generated by MWT (c) Another feasible schedule

2.2 Motivation

The motivation behind this work is that existing sleep scheduling for multiple CBR connections do not aim at minimizing state transitions. Schedules produced by these schemes can thus be further optimized to save more energy.

As an demonstration, consider four coexisting CBR connections with parameters shown in Table 1. If the MSS is allocated a frame capacity that is large enough to accommodate five packets in a frame, MWT will generate a five-frame sleep schedule with sleep ratio 0.4 as shown in Fig. 2(b). This schedule gives rise to two state transitions (counting only sleep-to-active transitions). However, packet transmissions in the second and third frames of the schedule can be postponed to the third and fourth frames, respectively, without violating the delay constraint of any connection. See Fig. 2(c). Such reschedule reduces state transitions while preserving the same sleep ratio.

This rescheduling is possible since MWT attempts to transmit packets in the earliest possible frame that maximizes bandwidth utilization to minimize packet delivery latency. This strategy leaves room for deferring transmissions. Deferring transmissions may not be feasible for other scheduling schemes such as AS, which tend to transmit packets in the latest possible frame within the delay constraint to maximize the likelihood

of transmitting multiple packets in a single frame. For these schemes, rescheduling is still possible by shifting transmission schedules backward rather than forward. In that case, packet delivery latency is effectively reduced due to accelerations of packet transmissions by the rescheduling. Whichever rescheduling policy is taken, it is not trivial to find a feasible reschedule that minimizes state transitions yet still respects packet order and delay constraints.

Energy saving by state transition reductions is significant when the ratio of the startup time to the transmission time is sizable [13]. However, the exact amount of energy saved varies, depending on hardware characteristics of the transceiver in use and the implementation of the sleep mode with the transceiver. For example, many RF transceiver chips provide two or even more kinds of power-saving modes. A typical *shutdown* mode powers off all functional modules (transmitter, receiver, and local oscillator) within the chip to conserve energy to the most possible extent while a *standby* mode may keep the local oscillator on to track carrier frequency. If a system designer chooses standby mode in implementing MSS sleep function, the resultant startup time is negligible but power dissipation in the sleep mode may be significant. If in stead the shutdown mode is enabled during sleep time, the startup time can be only one order of magnitude lower than the transmission time. For example, the PLL (phase-locked loop) settling time[1] reported by a 2.3-2.7 GHz WiMAX transceiver is at least 0.5 ms [14]. This amount is one-fourth of the minimal 2-ms frame duration used by WiMAX, which is significant.

State transition reduction may also facilitate the realization of sleep schedules under IEEE 802.16e infrastructure. How to realize sleep schedules generated by scheduling algorithms is an issue not seriously addressed before. Some work [6], [7] implicitly assumes the application of a single Type II power-saving class. For schedules that exhibit simple periodicity (such as those shown in Fig. 2), it is feasible to seek a common listening interval to apply Type II power-saving class. By merging together separated active periods into consecutive one, the length of listening interval can be reduced and energy is saved. Refer to Fig. 3 for an example.

On the other hand, when coexisting connections significantly differ in QoS requirements and traffic patterns, the resultant schedule may be so irregular that the use of a single instance of Types II power-saving class will give rise to poor bandwidth utilization and thus low sleep ratio (Fig. 4(a)). A possible remedy is to activate multiple instances of Type II power-saving class, which together precisely depict the schedule without introducing unneeded listening intervals (Fig. 4(b)). In this case, state transition reduction lessens the number of active instances and thus signaling overhead.

1. It is needed when the transceiver wakes up from the shutdown mode.

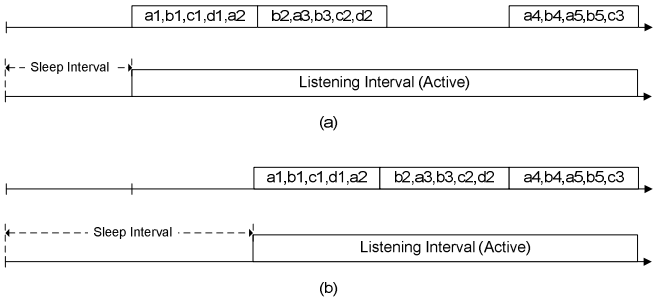

Fig. 3. Schedule realization using Type II power-saving class (a) for the schedule shown in Fig. 2(b) (b) for the schedule shown in Fig. 2(c)

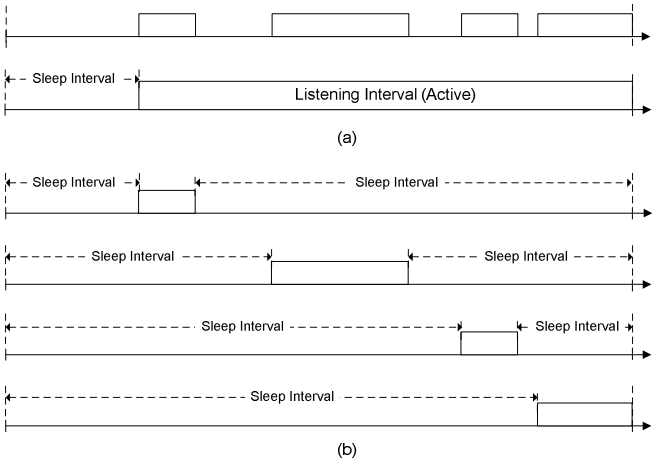

Fig. 4. Schedule realized by Type II power-saving class using (a) one instance and (b) four instances

3 Minimizing State Transitions

3.1 Principle

The basic idea behind the proposed approach is that packet transmissions in a given schedule can be accelerated or deferred to minimize the number of awake periods. The rescheduling, however, should meet the following criteria. First, packet transmission order in the original schedule should be preserved. Second, any deferral of packet transmissions should respect packet delay constraint that is specified as a QoS requirement. Third, the transmission of any packet cannot be accelerated to some time before the packet's arrival. We additionally demand that all packets originally scheduled in the same frame are still transmitted together in a frame after a rescheduling. This requirement avoids a complete re-scheduling that renders the original schedule useless. It also simplifies the rescheduling task significantly.

We first introduce some conventions to ease the ensuring discussion. Time in this section is measured by the duration of a frame unless otherwise specified. We assume all packets are of the same length, which is typical for streaming data. *Frame capacity* is the number of packets allowed for an MSS to be transmitted within the duration of a frame. A schedule is *feasible* if all

packets can be transmitted by the schedule without violating the QoS delay constraint of any connection.

For a packet p in a schedule, its *transmission shift*, denoted by $TS(p)$, is the time difference between the time p is generated and that p is scheduled to be sent. For a schedule to be feasible, every packet's transmission shift cannot exceed the packet's QoS delay constraint. The maximal additional transmission shift allowed for packet p in a schedule is defined to be the p's delay tolerance (DT) denoted by $DT(p)$. Intuitively, $DT(p)$ indicates how long we can further postpone p's transmission while still meeting its delay constraint.

Let $P(f)$ denote the set of packets scheduled to be sent in frame f. A frame f is *active* if $|P(f)| > 0$. For an active frame f, we define f's *maximum acceleration* (MA) to be

$$MA(f) = \min_{p \in P(f)} TS(p).$$

$MA(f)$ indicates the maximal number of frames that the scheduled transmission for all packets in $P(f)$ can be accelerated. Similarly, *maximum deferral* (MD) of a frame f is defined to be

$$MD(f) = \min_{p \in P(f)} DT(p).$$

It indicates the maximal number of frames that the transmission of all packets in $P(f)$ as a whole can be deferred without violating any packet's delay constraint.

For a sequence of consecutive frames $f_1, f_2, \cdots, f_m$, subsequence $f_i, f_{i+1}, \cdots, f_j$, where $1 \leq i \leq j \leq m$, constitute an *active group* if $i = 1$ or f_{i-1} is inactive, $j = m$ or f_{j+1} is inactive, and $f_i, f_{i+1}, \cdots, f_j$ are all active. All initial active groups can be found by an one-pass scan on all frames starting from the first. The first active frame encountered or the first active frame after an inactive frame defines the starting point of an active group, while the last active frame after the starting point concludes the active group.

By definition, two consecutive active groups are separated by one or more inactive frames. For an active group, its MA (resp. MD) value is defined to be the minimum MA (resp. MD) value of all frames in it. It is not hard to see that for any schedule, the number of active groups determines the number of state transitions. We may reduce the number of active groups by merging two separated active groups into a consecutive one by either accelerating the latter one or deferring the former one. There are, however, many ways to merge three or more active groups. Fig. 5 shows all possible ways to merge three active groups.

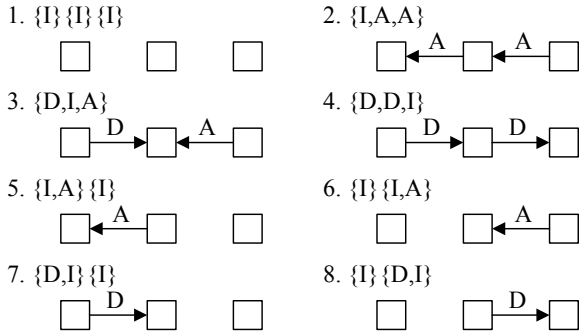

Fig. 5. Possible ways to merge three active groups. Groups merged together are enclosed by parentheses (A: acceleration; D: deferral)

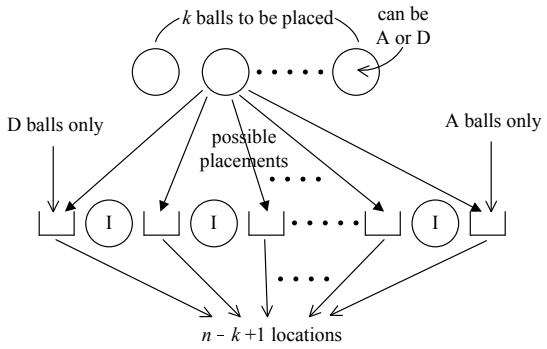

Fig. 6. Ball placement model for applying k group-merging operations to a schedule consisting of n active groups

3.2 Complexity Analysis

We shall now analyze the number of possible results after applying k group-merging operations (deferrals or accelerations) to a schedule consisting of n active groups. Let it be $p(n, k)$. Observe first that after k merging operations, where $k \leq n - 1$, k original active groups have been accelerated or deferred while the rest remain intact. We can model groups as balls and label them as 'A' (for those that are accelerated), 'D' (for those that are deferred), or 'I' (for all others), and each possible merging result corresponds to a possible permutation of placing k balls labeled either 'A' or 'D' into $n - k + 1$ possible locations between $n - k$ balls labeled 'I'. Each location has room for zero to k balls, but not every location can contain the same combination of balls. For example, the first location can only contain balls labeled 'D' while only balls labeled 'A' can be placed in the last location (Fig. 6).

When $k = 1$, there is one ball to be placed in one out of n locations. Except for the first and the last locations, where only one type of ball can be placed, all other $n - 1$ locations can contain both types of balls. The total number of permutations is therefore

$$p(n, 1) = 2 + 2(n - 2) = 2n - 2.$$

When $k \geq 2$, we denote a permutation of balls placed in the same location by their respect labels enclosed by brackets. For example, $[A, D]$ denotes two balls placed in the same location with the first labeled 'A' and the last labeled 'D'. When $k = 2$, two balls are to be placed in $n - 1$ locations. There are two possible cases:

- These two balls are placed in the same location. There can be only one possible ball permutation

when these two balls are both placed in the first ($[D, D]$) or in the last location ($[A, A]$). When these two balls are placed together in other locations, there are three possible ball permutations: $[A, A]$, $[A, D]$, and $[D, D]$. Ball permutation $[D, A]$ is excluded since we are required to preserve packet transmission order in the original schedule. The number of permutations in this case is therefore

$$2 + 3(n - 3) = 3n - 7. \quad (1)$$

- These two balls are placed in different locations. When one ball is placed in the first location and the other is placed in the last, only one ball permutation is possible (the first ball must be labeled 'D' while the last 'A'). When one ball is placed in the first location and the other is not placed in the last location, there are $2(n-3)$ possible ball permutations since there are $n - 3$ locations excluding the first and the last, and a ball in each possible location can be labeled 'A' or 'D'. The same situation applies to the case when one ball is placed in the last location and the other is not placed in the first location. The number of ball permutations when one or both balls are in the first or the last location is therefore

$$1 + 2 \times 2(n - 3) = 4n - 11. \quad (2)$$

  When $n > 3$ and neither ball is in the first or the last location, these two balls are to be placed in two out of $n-3$ locations. For each possible combination of locations, four permutations of ball labelings are possible. Therefore, the number of permutations is

$$4 \times \binom{n-3}{2} = 2n^2 - 14n + 24. \quad (3)$$

Summing up (1) to (3), we obtain the number of possible results after applying 2 group-merging operations to a schedule consisting of n active groups, which is

$$p(n, 2) = 2n^2 - 7n + 6.$$

The analysis for $k > 2$ is analogous but more complicated. We do not present the detail here due to space limitation but remark that for $n \geq 4$,

$$p(n, 3) = \frac{4}{3}n^3 - 16n^2 + \frac{236}{3}n - 140.$$

The total number of possible results after applying any number of merging operations on a schedule consisting of n groups is

$$\sum_{k=1}^{n-1} p(n, k).$$

In general, the number of permutations after applying k group-merging operations to a schedule consisting of n active groups is

$$\binom{n-k-1}{k}.$$

Therefore, the number of permutations is at least $n^{\frac{n}{2}}$ (occurring when $k = \frac{n-k-1}{2}$).

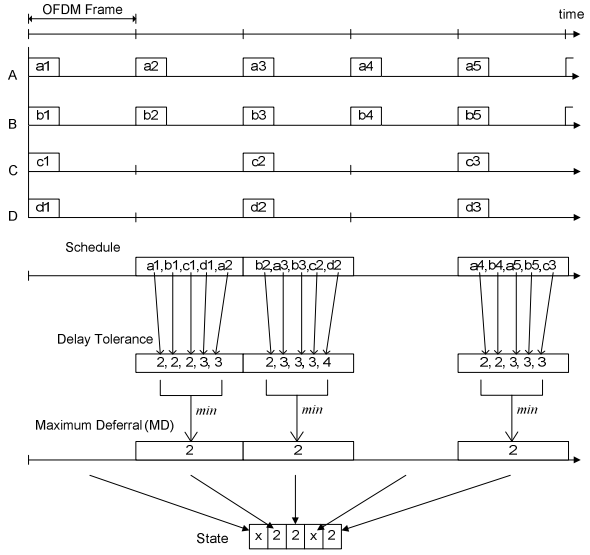

Fig. 7. State representation of the schedule shown in Fig. 2(b)

3.3 Proposed Scheme

In general, it is not trivial to generate all possible merging patterns for a schedule consisting of multiple active groups. Fortunately, we could significantly reduce the merging patterns by observing properties of the primary scheduler. For example, some scheduler like MWT tends to place packets in the earliest possible frame, so we need only consider possible deferrals of scheduled transmissions in rescheduling MWT's results. On the other hand, scheduler like AS prefers the latest possible frame within the delay constraint, so the schedule generated by AS can only be accelerated. In other words, we can consider only one possible direction on merging active groups.

We formulate the problem of converting a given feasible schedule into another with fewer state transitions as a state exploration problem. Our formulation demands that each schedule should have a unique state representation that contain adequate technical details for the rescheduling task. For a sleep schedule spanning n frames, we define its *state* to be an n-tuple, where the i-th element of the tuple is the MD value of the i-th frame. For any frame that is not active, the corresponding element in the tuple is simply marked with a '$\times$'. Fig. 7 illustrates how the state corresponding to the schedule of Fig. 2(a) is defined.

We propose two state exploration algorithms which consider only one type of merging operation (either deferral or acceleration). In the following, we present only deferrals of scheduled transmissions. Accelerating scheduled transmissions is analogous. The first algorithm termed OPA (One-Pass Adjustment) derives a single state from the initial state in a greedy manner. OPA scans all elements in the initial state from left to right, and puts off all active groups encountered to the most possible extent. By repeatedly merging together two adjacent active groups, OPA effectively reduces state

```
/* MD(G_i) denotes the MD value of active group G_i */
/* dist(G_i, G_j) denotes the distance between G_i and G_j */

proc OPA(state s)
    k ← the number of active groups in s
    i ← 1
    repeat
        let G_1, G_2, ..., G_k be the active groups of s
        if MD(G_i) ≥ dist(G_i, G_{i+1}) then
            merge together G_i and G_{i+1}
            s ← the new state after the merging
            k ← the number of active groups in s
        else
            i ← i + 1
        end if
    until (i = k)
    output s
end proc
```

Fig. 8. Algorithm OPA

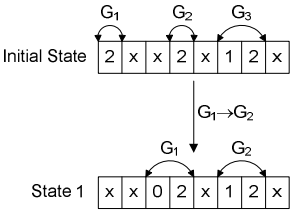

Fig. 9. Result of running OPA($\langle 2, \times, \times, 2, \times, 1, 2, \times \rangle$)

transitions.

OPA first looks for the first active group in the state, say G_i, and then determines whether G_i can be put off and merged into the next active group G_{i+1} by comparing the MD value of G_i and the distance between G_i and G_{i+1}. If the former is equal to or larger than the latter, G_i is allowed to be merged into G_{i+1}, forming a new active group with the same ordinal number G'_i. If G'_i can be formed, OPA updates MD values for all frames in G'_i that come from G_i, and examines G'_i for possible group merging with the group next to G'_i. Otherwise, OPA skips G_i and examines G_{i+1} instead. The examination ends when the second last active group has been checked. Fig. 8 shows the algorithm of OPA and Fig. 9 illustrates the result of running OPA(s), where $s = \langle 2, \times, \times, 2, \times, 1, 2, \times \rangle$. Observe that OPA successfully reduces the number of state transitions from three to two.

The second state exploration algorithm we proposed is ES (Exhaustive Search). As its name suggests, ES aims at exploring *all* feasible schedules that are derivable from the initial state (by performing one type of merging operation). In contrast, OPA yields only one schedule. Fig. 10 shows a recursive version of ES and Fig. 11 illustrates the result of running ES with the same initial state s. Among all states that are generated by ES, we pick up the one that has the minimal state transitions.

Note that ES does not actually explore all possible merging results discussed in the previous subsection.

```
/* MD(G_i) denotes the MD value of active group G_i */
/* dist(G_i, G_j) denotes the distance between G_i and G_j */
/* dist(G_i, ε) denotes the distance from G_i to the end of the
   schedule. */

proc ES(state s, int j)
    let G_1, G_2, ..., G_k be the active groups of s
    if j = k then
        if MD(G_k) ≥ dist(G_k, ε) then
            let s' be the state after moving G_k to
                the end of the schedule
            output s'
        end if
        return
    end for
    for i ← j to k do
        if MD(G_i) ≥ dist(G_i, G_{i+1}) then
            let s' be the state after merging together
                G_i and G_{i+1}
            output s'
            ES(s', i)
        end if
    end for
end proc
```

Fig. 10. Algorithm ES

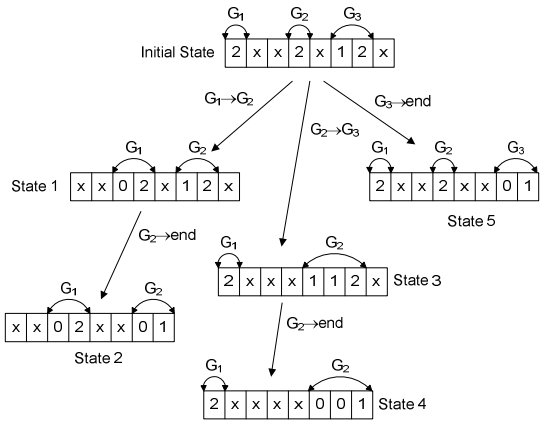

Fig. 11. Result of running ES($\langle 2, \times, \times, 2, \times, 1, 2, \times \rangle, 1$)

States that can only be formed by applying the other merging operation or by a mixed application of both operations are not examined. In fact, states examined by ES are much fewer than the half of all possible states. For an initial schedule consisting of four active groups, Fig. 12 illustrates how total $\sum_{k=1}^{3} p(4,k) = 20$ possible states can be derived by stepwise group-merging operations. We can see that ES explores only 7 out of 20 states (excluding the initial state). OPA explores even fewer states. If all the states examined by ES and all the attempts that derive these states are viewed as a tree rooted at the initial state, OPA explores only one path of the tree that leads to some leaf node[2].

In addition to all possible mergings between adjacent active groups, ES also attempts moving the last active

2. The leaf node is the only state that OPA returns.

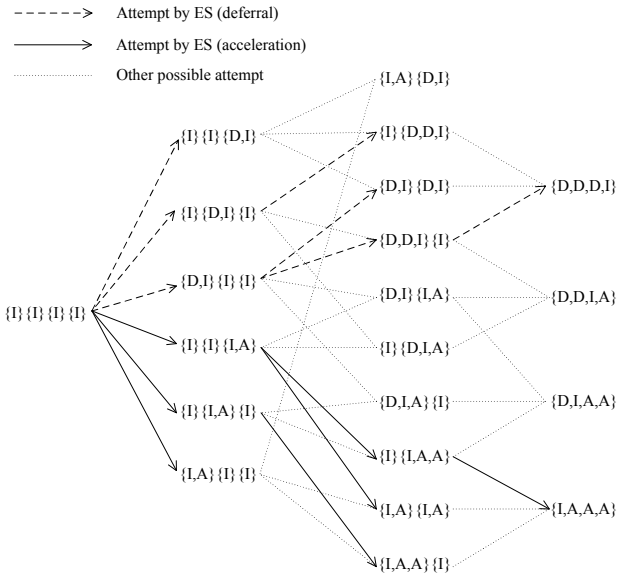

Fig. 12. Possible merging attempts and results for a schedule consisting of four active groups

group to the end of the schedule[3]. This attempt aims at further reducing state transitions when the resultant schedule exhibits periodicity. In that case, no state transition occurs between the last active group of the current scheduling period and the first active group of the next scheduling period if there is no inactive frame in between. Taking account of this wrap-around effect can remove one more state transition within a schedule period form a long-term perspective. In Fig. 11, state 4 will be chosen if the wrap-around effect is taken into account. OPA needs not consider the wrap-around effect as it seldom leaves active groups on both ends of the schedule.

To really enjoy the benefit of the wrap-around effect, the given schedule should be periodic and a state in our method should represent a whole period of the given schedule. However, not every schedule exhibits periodicity. For example, AS with certain range of parameter settings may yield aperiodic schedules. For this reason, the wrap-around effect is only considered a tie-breaking metric to deal with the case when ES finds multiple states having the minimal number of state transitions.

It should be noted that even for periodic schedules, a period-wide state representation may still lead to a suboptimal solution. Refer to Fig. 13 as an example, where a periodic schedule yielded by MWT is given for rescheduling. It can be seen that OPA's reschedule with a state spanning a whole period yields one more state transition than that with a state spanning two periods. It is difficult, if not impossible, to find the optimal number of periods that a state stands for to minimize the number of state transitions. Generally speaking, the number of state transitions tends to be lower when the number of

3. For the acceleration version, ES attempts moving the first active group to the front of the schedule.

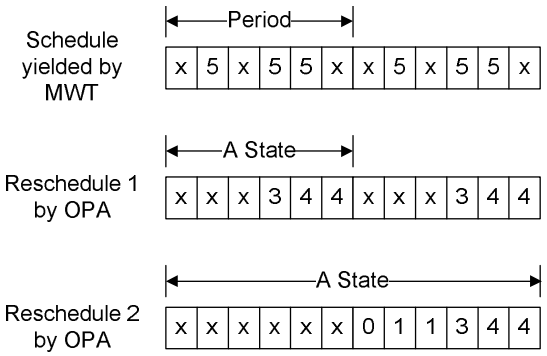

Fig. 13. Different rescheduling results with different state representations

TABLE 2
Parameter setting

Parameter	Value/Range
Frame duration	5 ms
Frame capacity	5 packets
Number of connections (n)	1 to 10 (default 4)
Inter-packet arrival time (λ)	5 to 30 ms
QoS delay constraint (d)	30 to 150 ms
Simulation time	500 ms

periods that a state represents is higher. We demonstrate this tendency by experiments in the next section.

When a state comprises many active groups, time complexities of the proposed algorithms may be a concern. Since group merging is the most time consuming operation in the proposed algorithms, we count only the number of mergings in the following time analysis. Let m be the number of frames in an initial schedule. Since adjacent active groups must be separated by at least one inactive frame, at most $\lceil \frac{m}{2} \rceil$ active groups can be defined. For a state that consists of n elements, at most $n-1$ mergings can be made by OPA. Therefore, the time complexity of OPA is $O(\lceil \frac{m}{2} \rceil - 1) = O(m)$. ES tries all possible combinations of group movements, so the worst case occurs when every active group in the initial schedule can be moved (deferred or accelerated). In that case, total $2^{\lceil \frac{m}{2} \rceil}$ states (including the initial state) are examined by ES. The time complexity of ES is therefore $O(2^m)$.

4 Performance Evaluation

We conducted simulations to investigate the performance of the proposed schemes. Both MWT [2] and AS [6] were used to yield initial sleep schedules. OPA and ES were then applied to reschedule the initial schedules. For initial schedules produced by AS, OPA and ES work by shifting transmission schedules backward rather than forward. We then compared the results among all possible treatments.

Table 2 lists the parameter setting for the simulations. The application under consideration is VoIP using PCM [15], [16] or ADPCM [17], [18] coding scheme, which

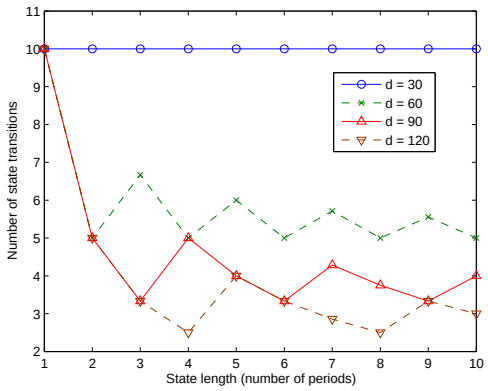

Fig. 14. Number of state transitions per 100 frames after running OPA on a schedule yielded by MWT with $n = 4$ and $\lambda = 10$ ms

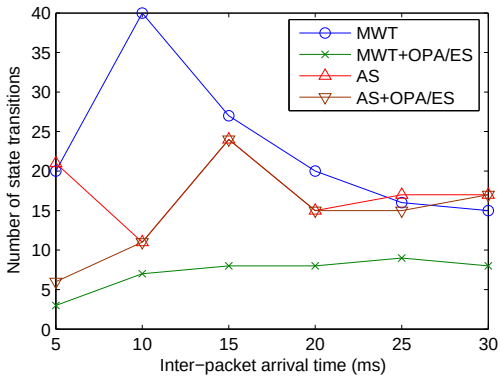

Fig. 15. Number of state transitions (within 500 ms) versus λ ($n = 4, d = 50$ ms)

yields a data rate of 64, 40, 32, 24, or 16 Kbps per session (connection). We assume fix-sized packets. With inter-packet arrival time (λ) ranging from 5 to 30 ms, the sizes of packets range from 80 bits/packet (16 Kbps ADPCM, λ = 5 ms) to 1920 bits/packet (64 Kbps PCM, λ = 30 ms). The frame capacity of WiMAX is flexible and we assume that five packets can be properly contained in a frame. Delay constraints of packets do not exceed 150 ms, as recommended by ITU-T [19]. The frame duration (5 ms) is typical in WiMAX.

4.1 Number of State Transitions

We first studied how the proposed approach reduces the number of state transitions. As we have mentioned, the results may depend on the length of a state. Fig. 14 shows the result of running OPA on a schedule yielded by MWT with various settings of state length. The initial schedule exhibits four state transitions in a period of 50 ms. OPA reduces the number to one state transition within a period. However, when the QoS delay constraint (d) is too tight to allow transmission deferrals across periods, a large state length does not help further reduce state transition times. With a large d, results exhibit oscillatory behaviors with the tendency of fewer state transitions in case of larger state length. Despite of the correlation between state length and state transition times, in the following simulations we simply take the results yielded by MWT as the input to OPA/ES for fair comparisons.

We next fixed the number of connections (n) and the QoS delay constraint (d), and observed how the number of state transitions changed with the setting of inter-packet arrival time (λ). Fig. 15 shows the result. As the results of OPA and ES are hardly distinguishable, we present only one set of data for them termed "OPA/ES". It can be seen that both MWT's and AS's results are affected by λ. For MWT, OPA/ES's improvements are significant: the resultant number of state transitions is much lower than that with MWT and only slightly increases with increasing λ. For AS, OPA/ES's improvements are significant with $\lambda = 5$ but not much with other settings. Although AS generally outperforms MWT, MWT+OPA/ES exhibits even fewer state transitions than AS+OPA/ES.

The result also reveals that OPA/ES's performance is not improved with increasing λ. This can be explained as, for an enlarged λ, packet's transmission shift should be increased (as long as d allows) to cluster together successive packets. The increase of a packet's transmission shift decreases the MD value of the associated frame, which implies a decreasing probability of clustering together two adjacent active groups (provided that the distance between consecutive active groups in the initial state does not change significantly with increasing λ).

We performed another set of simulations to investigate the relationship between state transition times and d. We made d a multiple of 15 ranging from 30 to 150. Considering the fact that every connection may have different delay constraint in the real world, we randomly chose d for each connection from the range $[30, d_{max}]$, where d_{max} denotes the upper bound of d. The value of d_{max} bears two meanings. It can be thought of a gauge of divergence of d among connections. Besides, a large value of d_{max} also indicates a high expected value of d.

Figures 16 and 17 show the results for MWT and AS, respectively, with increasing d_{max} and different λ. We observed that for a specific λ value, MWT's results are independent of d_{max}. This is reasonable as MWT tends to schedule packet's transmission in the earliest possible frame, so a loose delay constraint does not matter. OPA/ES's improvements on MWT are significant with small λ and large d_{max}. This can be justified as for MWT, a small λ leads to small transmission shifts while a large d allows for large values of MD. AS's results do not relate to d_{max} when λ is small, so OPA/ES significantly improves AS for the same reason as OPA/ES improves MWT. When λ is large enough, AS may cluster together packet transmissions with large d_{max}. Consequently, the number of state transitions is reduced. Nevertheless, OPA/ES still improves AS, particularly when d_{max} be-

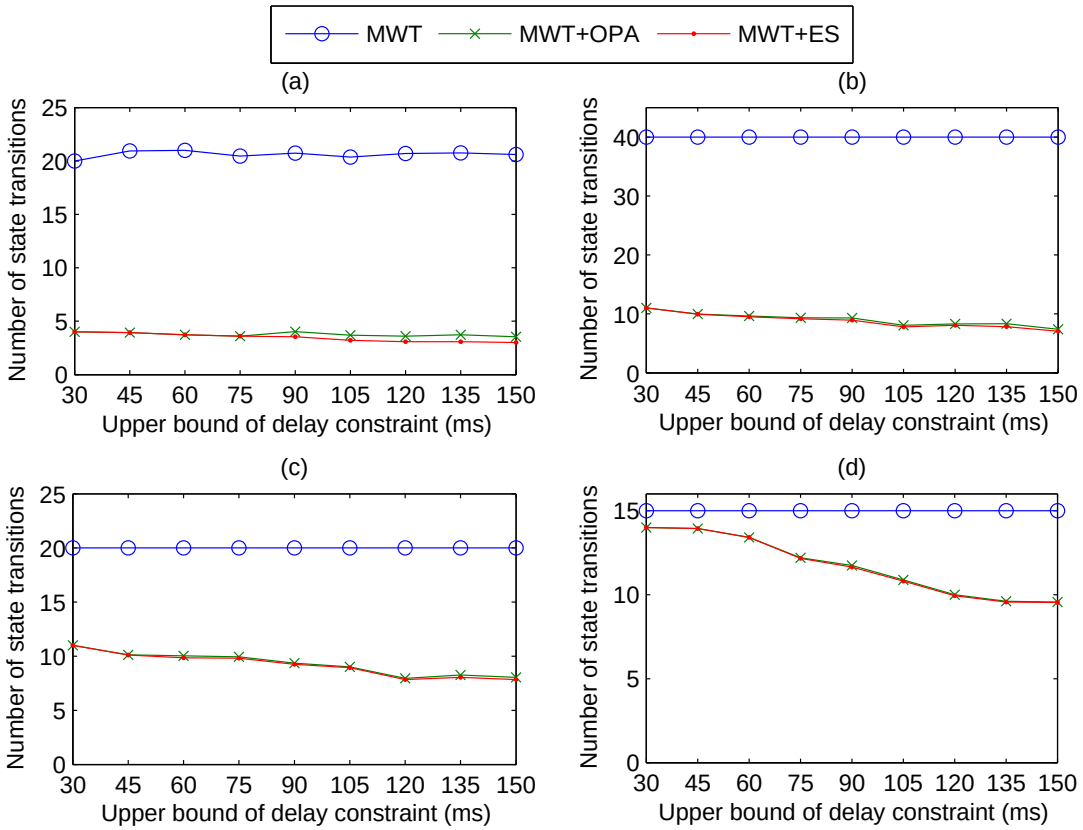

Fig. 16. Number of state transitions versus the upper bound of delay constraint in MWT (a) $\lambda = 5$ ms (b) $\lambda = 10$ ms (c) $\lambda = 20$ ms (d) $\lambda = 30$ ms. ($n = 4$)

comes large.

Figure 18 shows the number of state transitions versus fixed d with λ of each connection randomly chosen from $\{5, 10, 15, 20\}$. We observed that the diversity of λ leaves more room for OPA/ES to improve. This is particularly significant for the case of AS.

We also investigated the relationship between n and the number of state transitions. Fig. 19 shows the result. Here the performance of MWT+OPA/ES is stable and independent of MWT while that of AS+OPA/ES is closely related to AS. The result of MWT deserves further explanation. When $n < 5$, each new connection causes MWT to create a new active group, where adjacent active groups are separated by one inactive frame. That is why the number of state transitions increases linearly with n when $n \leq 5$. After five active groups have been creased, each new connection takes an inactive frame between two adjacent active groups, effectively merging together these two active groups. Consequently, the number of state transitions decreases linearly when $5 < n \leq 10$.

4.2 Packet Delay

The proposed schemes affect packet delay in two ways. Packet delay is increased when OPA/ES is used to postpone packet transmissions of the original schedule. On the other hand, packet delay can be decreased in case of packet transmission acceleration. Fig. 20 shows average packet delay for MWT-based schedules. This result, together with that shown in Fig. 16, indicates that the improvement on MWT's state transition times comes at the cost of additional packet delay. The cost is acceptable since all resultant delays are still under the constraint. Average packet delays for AS-based schedules are shown in Fig. 21. It can be seen that the proposed schemes can improve both state transition times and packet delays for AS, particularly with small λ and large d. Notably OPA/ES does not add extra packet delay to AS's schedules in all settings.

5 Conclusions

We have proposed two schemes, OPA and ES, to further reduce state transitions of a given sleep schedule. These two schemes work by accelerating or deferring scheduled packet transmissions. We have done extensive simulations to investigate the performance of OPA and ES. The results indicate that the performance of OPA and ES is hardly distinguishable in most cases. Both improve AS and MWT in the number of state transitions. The improvements on MWT are, however, more significant than those on AS. For MWT, the proposed schemes incur additional packet transmission delay. For AS, the proposed schemes generally decrease packet transmission delay. In short, OPA is recommended as a light-

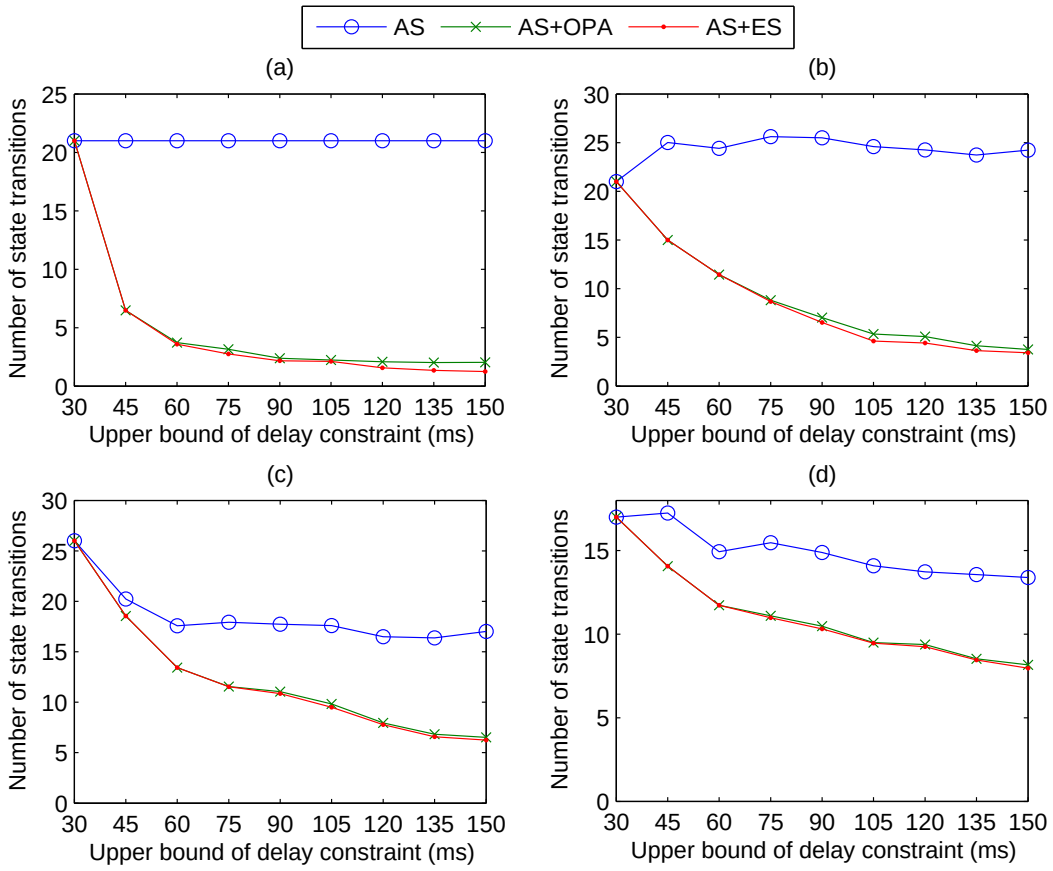

Fig. 17. Number of state transitions versus the upper bound of delay constraint in AS (a) $\lambda = 5$ ms (b) $\lambda = 10$ ms (c) $\lambda = 20$ ms (d) $\lambda = 30$ ms. ($n = 4$)

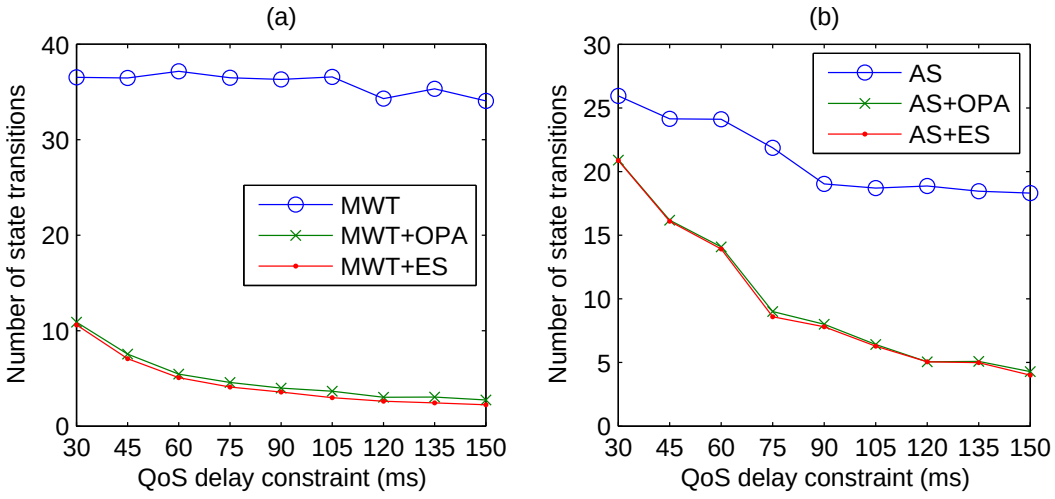

Fig. 18. Number of state transitions versus d ($n = 4$, λ: randomly chosen from $\{5, 10, 15, 20\}$) (a) MWT-based results (b) AS-based results

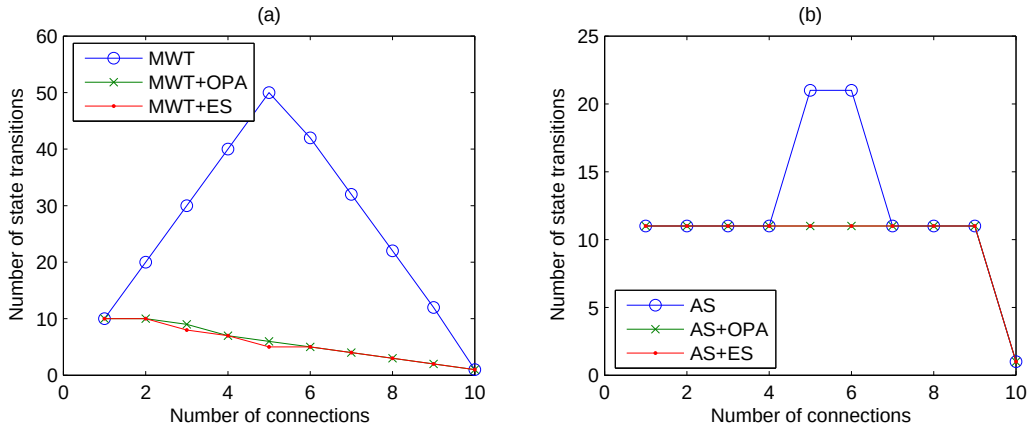

Fig. 19. Number of state transitions versus n ($\lambda = 10, d = 50$ ms) (a) MWT-based results (b) AS-based results

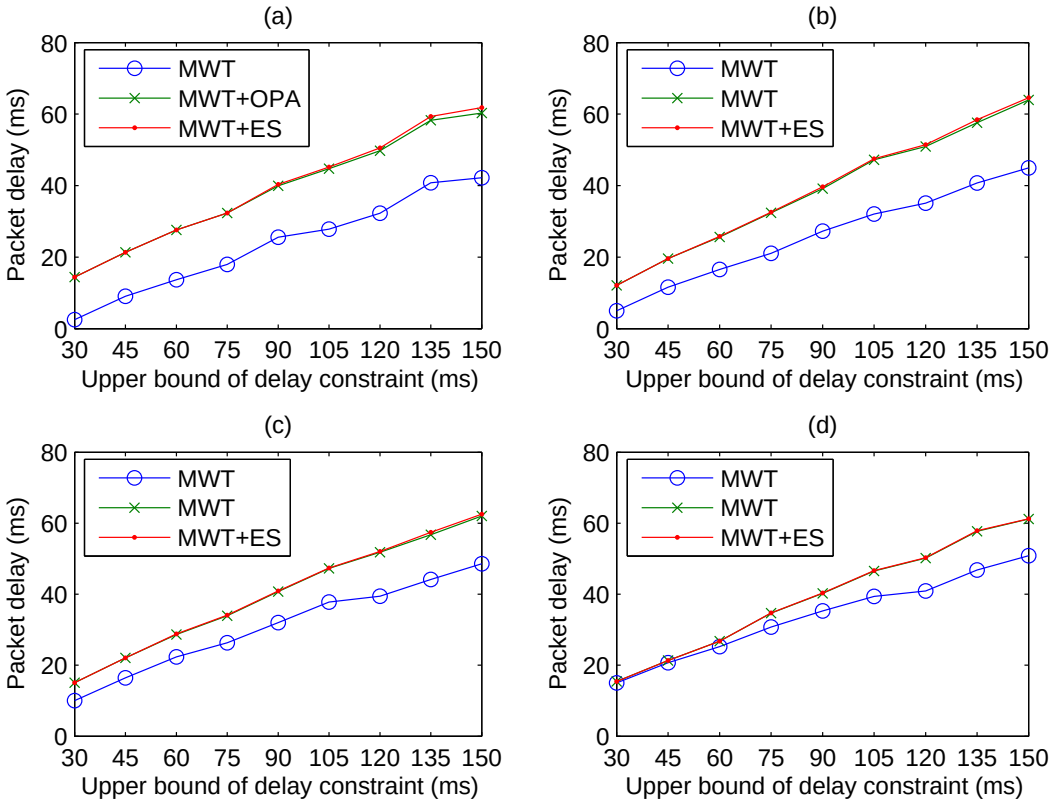

Fig. 20. Average packet delay versus the upper bound of delay constraint in MWT (a) $\lambda = 5$ ms (b) $\lambda = 10$ ms (c) $\lambda = 20$ ms (d) $\lambda = 30$ ms. ($n = 4$)

weight augmentation to sleep scheduling schemes for state transition reduction.

Reducing state transitions by merging active groups of an MSS may also complicate the task of scheduling transmissions for multiple MSSs. How to reduce state transitions for transmissions from multiple MSSs is a difficult issue and deserves a future study.

REFERENCES

[1] *Air Interface for Fixed and Mobile Broadband Wireless Access Systems*, IEEE Std. 802.16e, Dec. 2005.

[2] S.-C. Huang, R.-H. Jan, and C. Chen, "Energy efficient scheduling with QoS guarantee for IEEE 802.16e broadband wireless access networks," in *Proc. 2007 Intl Conf. on Wireless Communications and Mobile Computing*, Honolulu, Hawaii, USA, 2007, pp. 547–552.

[3] J. Shi, G. Fang, Y. Sun, J. Zhou, Z. Li, and E. Dutkiewicz, "Improving mobile station energy efficiency in IEEE 802.16e WMAN by burst scheduling," in *Proc. IEEE GLOBECOM*, 2006.

[4] L. Tian, Y. Yang, J. Shi, E. Dutkiewicz, and G. Fang, "Energy efficient integrated scheduling of unicast and multicast traffic in 802.16e WMANs," in *Proc. IEEE GLOBECOM*, 2007, pp. 3478–3482.

[5] J. Jang, K. Hant, and S. Choi, "Adaptive power saving strategies for IEEE 802.16e mobile broadband wireless access," in *Asia-Pacific*

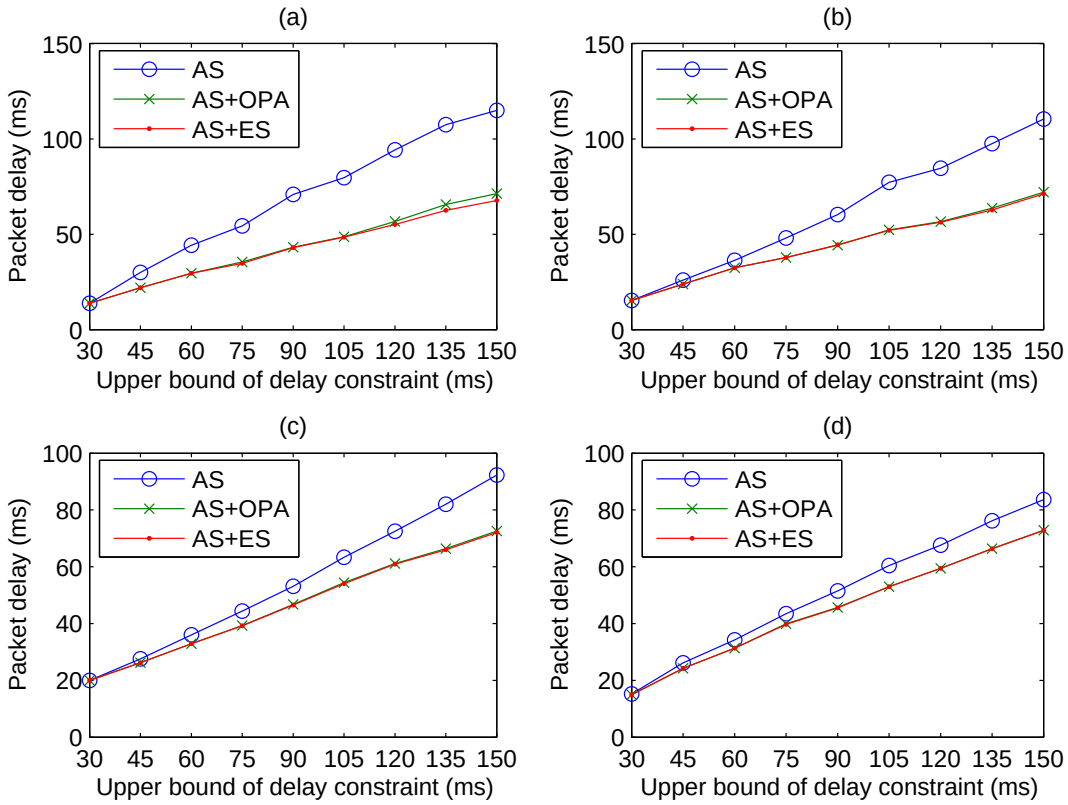

Fig. 21. Average packet delay versus the upper bound of delay constraint in AS (a) $\lambda = 5$ ms (b) $\lambda = 10$ ms (c) $\lambda = 20$ ms (d) $\lambda = 30$ ms. ($n = 4$)

Conference on Communications, 2006, pp. 1–5.

[6] S.-L. Tsao and Y.-L. Chen, "Energy-efficient packet scheduling algorithms for real-time communications in a mobile WiMAX system," *Computer Communications*, vol. 31, pp. 2350–2359, 2008.

[7] H.-L. Tseng, Y.-P. Hsu, C.-H. Hsu, P.-H. Tseng, and K.-T. Fen, "A maximal power-conserving scheduling algorithm for broadband wireless networks," in *Proc. IEEE WCNC*, 2008, pp. 1877–1882.

[8] J. A. Stine and G. de Vecian, "Improving energy efficiency of centrally controlled wireless data networks," *Wireless Networks*, pp. 681–700, 2002.

[9] R. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless web access with bounded slow-down," in *Proc. ACM/IEEE MobiCom*, 2002, pp. 119–130.

[10] J. B. Seo, S. Q. Lee, N. Park, H. Lee, and C. Cho, "Performance analysis of sleep mode operation in IEEE 802.16e," in *Proc. VTC 2004-Fall*, Sep. 2004, pp. 1169–1173.

[11] K. Han and S. Choi, "Performance analysis of sleep mode operation in IEEE 802.16e mobile broadband wireless access systems," in *Proc. IEEE VTC 2006-Spring*, May 2006, pp. 1141–1145.

[12] K. Lei and D. H. K. Tsang, "Performance study of power saving classes of type I and II in IEEE 802.16e," in *Proc. IEEE Intl Conf. on Local Computer Networks*, Nov. 2006, pp. 20–27.

[13] A. Y. Wang, "Low power RF transceiver modeling and design for wireless microsensor networks," Ph.D. dissertation, MIT, USA, Jun. 2005.

[14] (2008, Nov.) MAX2837: 2.3GHz to 2.7GHz wireless broadband RF transceiver. MAXIM. [Online]. Available: http://datasheets.maxim-ic.com/en/ds/MAX2837.pdf

[15] ITU-T Recommendation G.711, "Pulse code modulation (PCM) of voice frequencies," 1993.

[16] ITU-T Recommendation G.712, "Transmission performance characteristics of pulse code modulation channels," Nov. 1996.

[17] ITU-T Recommendation G.726, "40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (ADPCM)," 1990.

[18] ITU-T Recommendation G.727, "5-, 4-, 3- and 2-bits sample embedded adaptive differential pulse code modulation (ADPCM)," 1990.

[19] ITU-T Recommendation G.144, "One-way transmission time," May 2003.