802.11ax TWT Power Saving Mode Simulation

Patrick Hsu, *Undergraduate, NTUEE,* Bo-Ying Huang , *Undergraduate, NTUEE,*
and Chuang-Yu Chang, *Undergraduate, NTUEE*

***Abstract*—IEEE 802.11ax WLANs introduce Orthogonal Frequency Division Multiple Access (OFDMA) to improve throughput in dense scenarios. To save power of battery operated stations (STAs), a novel broadcast Target Wake Time (TWT) operation for negotiating wake time between an access point (AP) and a group of STAs is also proposed by making full use of the new capability of uplink OFDMA-based multiuser transmissions. But if the wake time of each STA is not properly scheduled, deteriorated throughput, high latency, and high power consumption occur because of collisions. In this paper, we refer to *"A Target Wake Time Scheduling Scheme for Uplink Multiuser Transmission in IEEE 802.11ax-Based Next Generation WLANs"*[1], with a few extra insights and modification. For example, we design a simple algorithm to find the optimal average number of station waking up in a beacon slot, and in turn get the optimal listen interval (LI). Then we use the optimized LI to determine the first target beacon transmission time(first TBTT). We can make a full schedule of time for waking up of any STA by knowing its first TBTT and LI. In the end, we compare the performance with original TIM triggered power saving method, and also shows how our algorithm improve this problem step by step.**

***Index Terms*—Target wake time (TWT), IEEE 802.11ax, power conservation, orthogonal frequency division multiple access (OFDMA).**

I. Introduction

NOWADAYS, some devices usually have little data (e.g. IOT devices) but have to keep power on even when they don't transmit data. As a result, there are some power management scheme in IEEE 802.11. For example, 802.11ax employs many new technologies like Orthogonal Frequency Division Multiple Access (OFDMA) based multiuser (MU) transmission, spacial reuse (SR), and Target Wake Time (TWT). When stations (STA) don't have data to transmit, they power down the transceivers, i.e. entering into power save mode (PSM). As shown in Fig. 1 [1], The STAs have to send TWT requirement to AP for knowing when to wake up to get target beacon containing the TWT offsets and wake intervals. The time to receive the target beacons is negotiated at the beginning through a Target Beacon Transmission Time (TBTT) negotiation procedure by setting the offset (i.e., the first TBTT) and interval of successive TBTTs (i.e., Listen Interval). Note that the initial TWT requirement contains the LI each STA calculates according how busy it is, so the LIs returned has been optimized by AP.

The key feature is 802.11ax makes use of the advantage of MU transmission capability. The STAs can send TWT requirement simultaneously and transmission time scheduled by AP is shared by multiple one STAs, unlike the individual TWT agreement [2]. In dense scenarios, many STAs may send TWT requirement during a short period, the AP has to assign the first TBTTs and LIs properly to prevent STAs from waking up simultaneously to receive the same target beacon and request for limited random access resource units (RA-RUs). The scheme for scheduling a proper TWT is out of scope in the IEEE 802.11ax and is implementation-specific. So we refer to a paper [1] proposing a TWT based sleep/wake-up scheduling scheme (TSS) to negotiate the first TBTT and LI while taking OFDMA based MU transmission capability into consideration. The LI is adaptive according to the network density and the number of available RA-RU. In addition, assigning different first TBTTs, we can avoiding the alignment of wake up time of STAs and decrease the number of simultaneously active STAs and alleviates average contention in each beacon slot. We would also compare the throughput and energy effiency (EE) of the network with and without TSS under different scenarios.

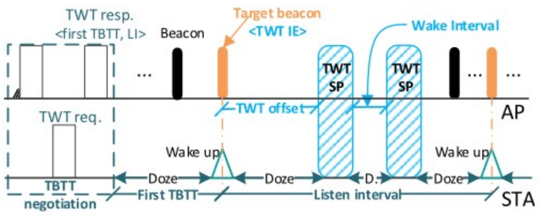


Fig. 1: An example of TBTT negotiation procedure

II. The optimization of LI

Before explaining the algorithm for the optimization of LI, we need to introduce a parameter $\overline{d_{cl}}$, which is average number of STAs waking up in one beacon slot (like frequency). We know LI is the how much time each STA enter into PSM again (like period), thus $\overline{d_{cl}}$ can be calculated as follows,

$$\overline{d_{cl}} = \sum_{i=1}^{n} \frac{1}{LI_i}. \tag{1}$$

, where LI_i is the LI of i_{th} station. We can get the optimal LI by finding the optimal $\overline{d_{cl}}$. Also, we have the total effective throughput θ on m RA-RUs from the following equation,

$$\theta = \overline{d_{cl}} \cdot p_{sb} \cdot p_{ru} \cdot r \cdot T_{effective}, \tag{2}$$

where p_{sb} is the probability of successful backoff, p_{ru} is the successful probability of choosing an idle RA-RU that is not selected by the other STAs, r is the bit rate, and $T_{effective}$ is ratio of the time really spent on transmitting data. p_{sb} and p_{ru}

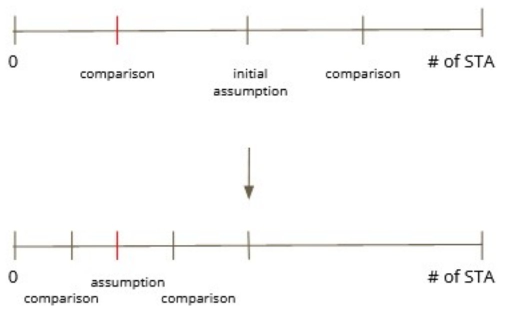


Fig. 2: The procedure of finding $\overline{d_{cl}}$

are negative related to $\overline{d_{cl}}$ and positive related to the number of RA-RUs m . The optimal $\overline{d_{cl}}$ is one which can maximize θ. As the upper bound of $\overline{d_{cl}}$ is the number of all STAs (i.e. all the STAs wake up in every beacon slot) and the lower bound is 0, we first compare θ when $\overline{d_{cl}}$ is $\frac{1}{4}$, $\frac{1}{2}$, and $\frac{3}{4}$ of the number of all STAs. If θ is the most when $\overline{d_{cl}}$ is $\frac{1}{4}$ of the number of all STAs, we choose it as the new mid and do the above procedure again for a few iterations as Fig. 2 shows.

When the number of STAs, the iteration is less than 10 and optimized $\overline{d_{cl}}$ is 12.48. After we have the optimized $\overline{d_{cl}}$, we can find optimized LIs by eq(1), i.e.

$$LI_i^{'} = \max\{R(t_i \cdot \overline{d_{cl}}/\overline{d_{cl}^{'}}), 1\} \tag{3}$$

where R denotes the rounding function to get an integer. If the solution of LI is 0, the smallest LI 1 is allocated to the STA.

III. The scheduling of First TBTT

After determining LI of the STAs, we can use the optimized LI on finding first TBTT to make contention level more equally distributed, which can substantially improve the performance. The algorithm we used to determine first TBTT have three parts: inter-grouping, intra-grouping, and drift-grouping.

A. Inter-grouping Algorithm

Inter-grouping algorithm will divide the STAs into different subsets according to their LIs. The dividing rule is that in every subsets, choosing two STAs arbitrarily will have a relationship of divisor and multiple between the LIs of the two STAs. Since if two STAs have LIs of relationship of divisor and multiple, their TBTT can be staggered if they do not wake up at the first TBTT of the other. In our simulation, we used the algorithm provided by Chen et al.(2019)[1]. Firstly sort the STAs by their LIs ascending. Check the STAs in order, for each STA if it can be divided into a existed subset then we do it, or we create a new subset.

B. Intra-grouping Algorithm

Intra-grouping algorithm will deal with the STAs in one subset. It schedule the the first TBTT of individual STA to make the contention level of each beacon slot as equal as possible by adding some offsets. The STAs will be placed in timelines, which have their each slots represent a beacon slot. In our simulation, we used the algorithm provided by Chen et al.(2019)[1]. Since in the inter-grouping algorithm we have already have each subset be in ascending order of LI, so we can simply do the scheduling: For each STA, we search from the first beginning of the timeline, if there's empty slot then we choose it as the first TBTT of this STA and mark the TBTTs(first TBTT + integer times LI) as occupied. If there's no empty slot, we create a new timeline.

C. Drift-grouping Algorithm

Drift-grouping algorithm will give offsets to those timelines which is not fully filled, which can prevent the high contention level exist at the very beginning of timelines due to the intra-grouping algorithm(we fill in the STAs from the front of timelines). Since there's no clearly implementation in the references, we implement it by ourselves. We firstly use greedy algorithm for each timeline, that is find a optimization of minimizing the difference between maximum and minimum contention level by exhaustive method whenever we deal with a new timeline. But we then found this is not efficient. So in the simulation we use uniform random variable to generate a start point, then we search from start point to find a empty slot in the previous timeline as the offset. By doing this we can get a good enough selection of offset efficiently.

IV. Simulation and Evaluation

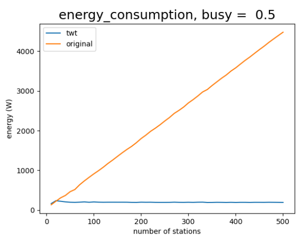
A. Simulation Setups

We use Poisson Distribution to model the packet arrival for each stations; all stations' lambda value are generated from a uniform distribution, the average lambda is calculated as follow:

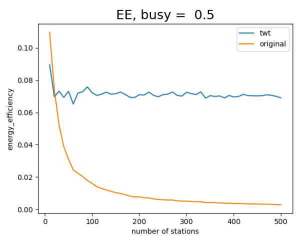
$$\lambda_{avg} = \frac{total\,channel\,capacity}{N_{STAs}} * r \tag{4}$$

r denotes the ratio of total traffic to the total capacity, which lies between 0 and 1. We assume that in each period of beacon, there are only m stations can get the channel to transmit, and other stations that didn't get the channel still need to keep awake in the same session, to simulate the random access process that they need kept trying until the time is over instead of fall back asleep. It is similar in the original TIM triggered power saving scenario, except the stations to wake is decided by whether it has something on TIM instead of the scheduled wake time. We ignored the periodical wake to listen the beacon in our simulation. Our default setting if m=8; data rate of each channel = 11.8 Mbps; session period = 0.1 ms, r = 0.5, $P_{tx} = 1W$, $P_{idle} = 0.3W$ and $P_{sleep} = 0W$. Also, we defined the following metrics: $throughput = average\,transmission\,rate$; $latency = average\,delay\,of\,packages$; $energy\,efficiency = \frac{data\,transmitted}{energy\,consumption}$.

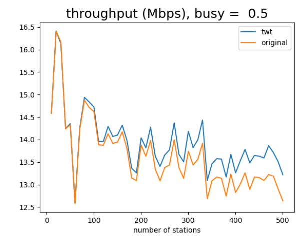
As shown in Fig. 3 there are significantly difference in energy efficiency and energy consumption metrics, and also outperformed in both throughput and latency under the setup of our simulation. The fluctuation in the figure is due to



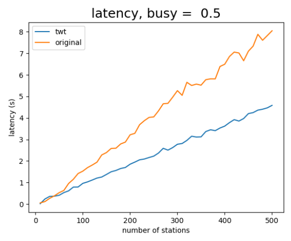
(a) energy consumption



(b) energy efficiency



(c) throughput



(d) latency

Fig. 3: performance compared to TIM triggered power saving mode

randomness when we are generating the stations, because these two methods are essentially very similar, thus if a set of station is bad/good for one, it would also bad for another. We also compared when m=4, shown in Fig. 4, the trend is very similar except the gap in through put is slightly larger.

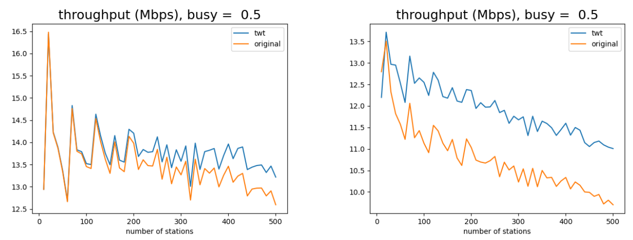
Finally, we separate our optimization procedure into three steps, which is ”station requested LI”, ”LI rescheduled” and ”first TBTT scheduled”, then evaluate the performance of these steps to know each operation’s impact. Fig. 5(a) shows that if we only rescheduled LI without scheduling the first TBTT, the performance is actually much worse than just apply request LI. The reason is because LI rescheduling algorithm is in brief scaling up the requested LI properly to adjust to the network status, but if we didn’t mismatch the beginning of each periodical wake, it is possible that many stations wake in the same session and sleep in other, and thus result in waste of channel resources. Meanwhile because the original requested LI is much smaller, there are always a lot of stations wake up, thus it’s more similar to randomly choose 8 stations among all, which at least is evenly distributed and has less waste. Fig 5. then shows a very positive result, after first TBTT scheduling, which is adjusting the first offset to reduce collision, both through put and latency improved dramatically. Although energy consumption increased, it is reasonable due to increased throughput, as long as our main goal, energy efficiency, has significant improvement.

Appendix A
Source Code

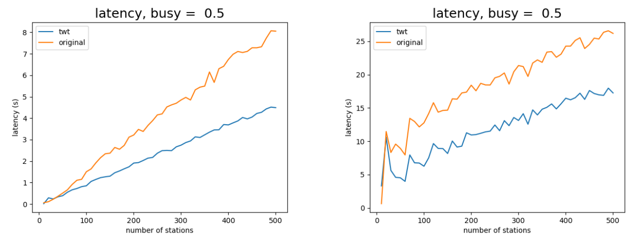
All our source code can be found at this github repository(https://github.com/x28150867/mobile-network-final-project).

References

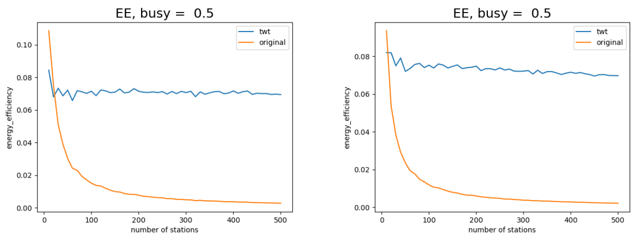
[1] Q. Chen, Z. Weng, X. xu and G. Chen, ”A Target Wake Time Scheduling Scheme for Uplink Multiuser Transmission in IEEE 802.11ax-Based Next Generation WLANs,” in IEEE Access, vol. 7, pp. 158207-158222, 2019, doi: 10.1109/ACCESS.2019.2950464.
[2] M. Nurchis and B. Bellalta, ”Target Wake Time: Scheduled Access in IEEE 802.11ax WLANs,” in IEEE Wireless Communications, vol. 26, no. 2, pp. 142-150, April 2019, doi: 10.1109/MWC.2019.1800163.



(a) throughput, right: m=8, left: m=4

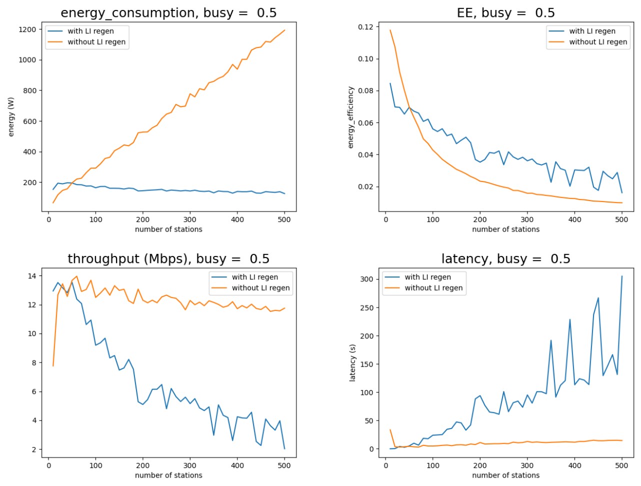


(b) latency, right: m=8, left: m=4

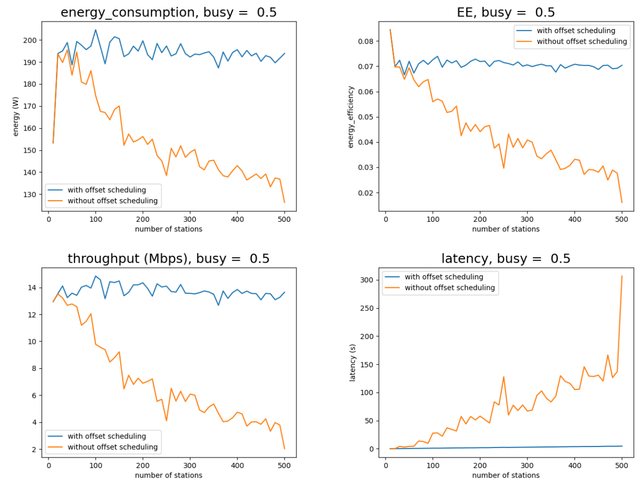


(c) energy efficiency, right: m=8, left: m=4

Fig. 4: performance under different number of channels



(a) performance with/without LI scheduling



(b) performance with/without first TBTT scheduling

Fig. 5: performance compared to TIM triggered power saving mode