IEEE Global Communications Conference
7-11 December 2020 // Taipei, Taiwan

Empirical Study and Enhancement of Association and Long Sleep in 802.11 IoT Systems

Simon Liu*, Vikram K. Ramanna*†, and Behnam Dezfouli*
*Internet of Things Research Lab, Department of Computer Science and Engineering, Santa Clara University, USA
†Infineon Technologies, San Jose, USA
{sliu3, vramanna, bdezfouli}@scu.edu

***Abstract*—The 802.11 standard, a.k.a., WiFi, is becoming more popular for IoT connectivity. The three essential operations performed to ensure connectivity in an 802.11 network are *association*, *maintaining association*, and *periodic beacon reception*. Understanding and enhancing the energy efficiency of these operations is essential for building IoT systems. Unfortunately, the overheads of these operations have not been studied considering station's software and hardware configuration, access point configuration, and link unreliability. In this paper, we show that: (i) association cost depends on multiple factors including probing, key generation, operating system, and network stack, (ii) increasing listen interval to reduce beacon reception wake-up instances may negatively impact energy efficiency, (iii) maintaining association by relying on the poll messages generated by the access point is not reliable, and (iv) key renewal aggravates the chance of disassociation. We also present station- and access point-based solutions that address some of these problems.**

***Index Terms*—Internet of Things, WiFi, Low Power, Association, Beacon.**

I. Introduction

Improving the energy efficiency of Internet of Things (IoT) systems is important from two perspectives: First, many of these devices (e.g., smart locks, security cameras) need to run on battery or rely on energy harvesting mechanisms. Second, the impact of the energy consumption of these devices on global Information and Communications Technology (ICT) energy footprint is increasing [1].

The 802.11 standard (a.k.a., WiFi) is an appealing technology for IoT connectivity [2]–[5]: First, large-scale deployment of 802.11 base stations, known as Access Points (APs), provides a ready infrastructure for IoT connectivity in license-free bands. Second, the high data rate of this standard compared to low-power technologies, such as 802.15.4 and Bluetooth Low Energy (BLE), is essential for applications such as medical monitoring, video streaming, process control, and robotics. Third, the energy consumption of 802.11 transceivers has been significantly reduced during recent years. Specifically, existing studies show the higher energy efficiency of 802.11's physical layer compared to BLE and LTE [4], [5].

IoT stations are resource-constrained and need to put their transceiver into sleep mode aggressively to reduce power consumption whenever no communication is taking place. However, regardless of the application type, all 802.11-based IoT systems need to perform the following operations to ensure AP-station connectivity: *association*, *maintaining association*, and *periodic reception of beacon packets*. Existing studies primarily focus on the effects of traffic on energy efficiency [2], [6], [7], or evaluate performance at a high level [8], [9], and less attention has been paid to the impact of the above operations on energy efficiency. To address this research gap, we present an empirical study of these operations and report factors that highly affect their overhead. We also propose solutions to improve the efficiency of these operations.

The contributions of this paper are as follows:

– Associating a station with an AP introduces non-negligible delay and energy consumption. We study the overheads of various association mechanisms using two different IoT stations, two operating systems, and two networking stacks. We also present solutions to reduce the identified overheads. For example, we show that employing Pairwise Master Key (PMK) offloading to IoT station's application processor reduces association duration and energy. Also, we show how programming the firmware to search for a particular AP can considerably reduce the overhead of polling. Another major observation is the impact of networking stack on association duration.
– A straightforward approach to reduce the overhead of beacon reception is increasing a station's listen interval. However, our results substantiate that this may result in a significantly longer wake-up duration to receive each beacon. We also show that the presence of downlink traffic or interference can further increase wake-up duration per beacon reception.
– To avoid the overhead of association and ensure seamless connectivity with AP, it is essential to maintain association with minimum overhead. This is particularly challenging because APs maintain a per-station inactivity timer to ensure the station is alive and within the communication range. Although APs by default poll inactive stations periodically, we prove that stations cannot rely on this mechanism to ensure connectivity. In particular, this may lead to *disassociation-unaware station*, which means the station is unaware of its disassociation by the AP. We propose station-generated keep-alive packets as a solution to this problem. We also confirm that the key renewal operations performed by AP are time-critical and may result in disassociation. We present customized AP configuration as a solution to this problem.

The rest of this paper is organized as follows: Section II overviews the power saving mechanisms of 802.11. Section III presents our research methodology. Association overhead is

studied in Section IV. In Section V we study the impact of listen interval on energy efficiency. Mechanisms for preventing reassociation are presented in Section VI. We overview the related work in Section VII. We conclude the paper in Section VIII.

II. Energy Saving Mechanisms in 802.11

The 802.11 standard offers various power-saving methods. The Power Save Mode (PSM) enables the stations to wake up periodically at each Beacon Interval (BI). The BI value used by commercial APs is 102.4 ms. Before transitioning into the sleep mode, the station informs the AP about its decision via a successful packet exchange where the power management bit (in the MAC header) is set to '1'. Every BI instance, the AP broadcasts a beacon message which, in addition to other information, conveys to the stations if the AP has buffered packets for the stations. This is performed by setting the Association ID (AID) bit of the station to '1' in the Traffic Indication Map (TIM) field of beacon. For each station, this is achieved by setting a particular bit called TIM to '1', if buffered packets are present. In this case, the station stays awake, sends a PS-Poll message to the AP, and waits for downlink packet transmission.

To deliver multicast and broadcast packets such as Address Resolution Protocol (ARP), the AP needs to ensure all the stations are awake. These packets are delivered after the transmission of a beacon packet including the Delivery Traffic Indication Message (DTIM) element. The DTIM period is configured as a multiple of beacon interval in the range 1 to 255. The number of beacons before the next DTIM beacon is conveyed by the *DTIM Count* field of each beacon. Therefore, a beacon packet with DTIM Count = 0 is a DTIM beacon.

With PSM, stations need to retrieve downlink packets every BI. Since the delay of this mechanism may not be acceptable for interactive applications, the Automatic Power Save Delivery (APSD) mechanism has been introduced. With APSD, a station can poll the AP anytime by sending a Null packet. The APSD mechanism, which is part of the 802.11e amendment (and is available in 802.11n/ac/ax), also introduces the concept of Access Category (AC), which differentiates the priority of voice (VO), video (VI), best effort (BE), and background (BK) traffic types. From left to right, the priority of channel access reduces.

III. Methodology

In this section, we present the common testbed configurations used in the experiments of this work.

The IoT stations used in our study are based on two low-power, widely-used 802.11 transceivers: CYW43907 [10], and BCM4343W [11]. CYW43907 (henceforth as the *CYW* station) includes two ARM-Cortex R4 processors. BCM4343W (henceforth as the *BCM* station) includes two ARM-Cortex M4 processors. In both systems, one core is dedicated to the 802.11 subsystem and the other core is used for running applications. These two transceivers are used by various low-power IoT development boards as well as products such as Amazon Tap, LG Urbane Watch, and Samsung Gear S2. Figure 1 shows the development boards that include these two transceivers. Unless otherwise mentioned, FreeRTOS and LwIP are, respectively, the real-time operating system and network stack used by the IoT stations.

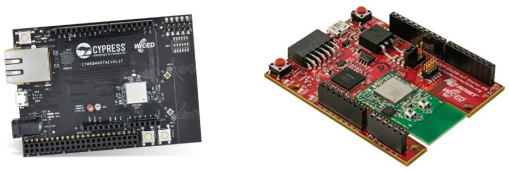


Fig. 1: The WiFi-based devices used in this paper. The left device includes CYW43907 and the right device includes BCM4343W. Both devices support FreeRTOS and ThreadX as their operating system. We refer to the left and right device as CYW and BCM, respectively.

The transceiver used by the AP is Atheros AR9462. AP functionality is handled by `hostapd`, which is a user-space daemon used by most commercial APs. This daemon is compatible with Linux-based systems and can be used to convert any AP-mode-supported transceiver into an AP. We use the 802.11n standard as the communication protocol between stations and AP. The AP operates in the 2.4 GHz band. The AP and the stations support PSM and APSD. Power saving is enabled on the stations for all the experiments.

The energy measurement tool is EMPIOT [12]. This platform's basic sampling rate is 500 Ksps, which are then averaged and streamed to the control software at 1 Ksps. Its maximum accuracy error is 4% compared to the existing high-end commercial products.

IV. Association Overhead

In this section, we analyze the overheads of association process and study solutions that reduce these overheads directly.

A. Association Process

The association process—a.k.a., *join*—is composed of the following steps. *Step 1*: The station scans for all nearby APs by sending probe requests on all the channels. The station collects all the responses, and among the APs whose SSIDs match with that programmed in the station, the compatible AP with the highest signal strength is chosen for the association. Here, compatibility refers to the matching of encryption algorithms and communication rates supported by the station and AP. *Step 2*: The station sends an association request to the AP and awaits to receive an association response. *Step 3*: The station performs key exchange with the AP. If no static IP has been configured, after step 3, the station needs to request for IP address using the DHCP protocol.

B. Association Mechanisms and Enhancements

We study the impact of the following parameters on association: probing, key offloading, operating system, network stack, and IP assignment. The station used in these experiments is CYW.

1) Probing: We use the following two join methods: (i) *Join*: The station performs all the three steps above. (ii) *Join Specific*: The station probes only a particular AP on a channel known to the station, then step 2 and step 3 are performed. The second method is implemented by using driver interfaces that allow us to program specific AP information in the firmware.

2) PMK Offloading: As per 802.11i enhancements, Robust Security Network (RSN) was introduced to perform four-way handshake and group key handshake. During the handshake, a shared secret key called PMK is generated. Using SHA-1, PMK is typically derived from the WiFi password, SSID, and SSID length. This function is implemented in the wireless driver, and therefore, it is run by the transceiver's processor. Since PMK generation is a process-intensive operation, offloading it to the application processor reduces association delay. This is possible for those IoT stations that utilize separate processors for controlling transceiver operation and running user applications. For example, the CYW station includes two processors: one for wireless connectivity, and one for user applications. By accessing low-level driver interfaces, we implement this mechanism on the CYW station by computing PMK in the application processor and then passing the computed value to the transceiver[1]. Considering PMK offloading, each of the two join methods mentioned above has two variations.

3) Operating System and Network Stack: We also consider the impact of *operating system* and *network stack*. The real-time operating systems used are FreeRTOS and ThreadX, and their networking stacks are LwIP and NetXDuo, respectively.

4) IP Assignment: For each of the above cases, depending on the configuration, the station either relies on DHCP or uses a statically allocated IP address.

C. Results and Discussion

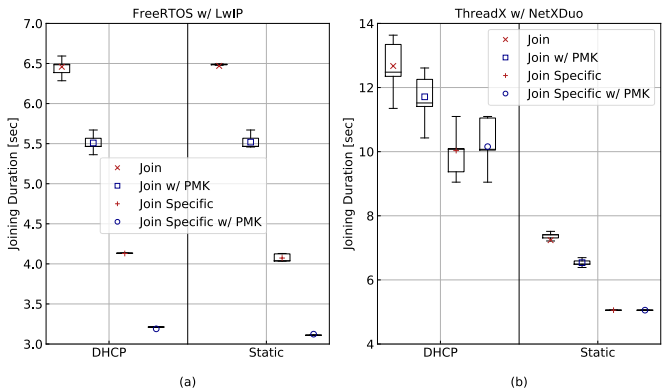
Figures 2(i) and (ii) present the duration and energy consumption of various join mechanisms, respectively. Using FreeRTOS w/ LwIP, we can observe that, on average, Join Specific reduces time and energy consumption by 39.66% and 43.58%, respectively, compared to Join. These values are 21.73% and 21.67% for ThreadX w/ NetXDuo. These results also justify the importance of PMK offloading. On average, using PMK offloading on FreeRTOS w/ LwIP reduces delay and energy consumption by 18.87% and 23.38%, respectively. These values are 4.01% and 5.39% for ThreadX w/ NetXDuo.

In terms of IP allocation, these results show the considerable impact of static IP allocation when using ThreadX w/ NetXDuo, compared to a marginal effect with FreeRTOS w/ LwIP. With NetXDuo, the join process includes the transmission of Gratuitous ARP packets after IP assignment. The station broadcasts these ARP packets to announce its IP to the entire network. This extra step is not performed by LwIP stack.

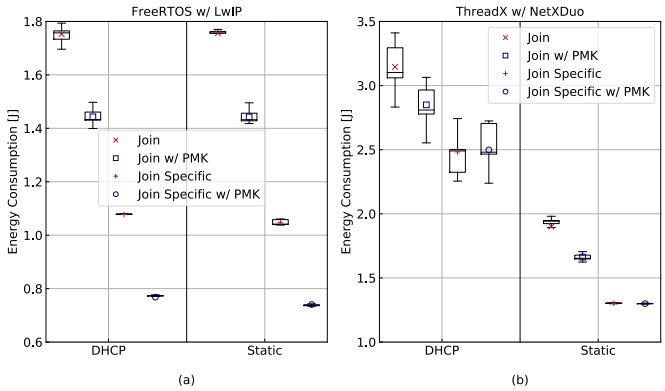
V. Beacon Reception Overhead

To reduce the number of periodic wake-ups for beacon reception, an IoT station can utilize a longer listen interval.

[1]Please use the following link to access the implementation: https://github.com/SIOTLAB/Low-Power-WiFi-Association.git



(i) Duration



(ii) Energy Consumption

Fig. 2: The overhead of various association (join) methods with CYW station. The PMK offloading mechanism is referred to as 'w/ PMK'.

The listen interval of a station is either conveyed to the AP at association/re-association time or during run time[2]. The *listen interval coefficient* (denoted as τ) specifies wake-up duration as a multiple of BI. For example, when $\tau = 10$, the station wakes up every 10×102.4 ms. In this section, we characterize the impact of listen interval, concurrent traffic, and interference on the wake-up duration of CYW and BCM to receive beacons. We measure the number of wake-ups to receive beacons as well as the time each station spends in awake mode before and after each beacon reception.

A. Configuration

We consider three cases: (i) normal, (ii) presence of *downlink* traffic, and (iii) presence of *interference*. In the normal case, the only station associated with the AP is CYW/BCM. In the downlink case, the AP is associated with the CYW/BCM station and a smartphone. The AP continuously transmits packets—that belong to the VO AC—to the smartphone. In the interference case, the AP is associated only with CYW/BCM. Also, another (similar) AP and a smartphone associated with it are operating nearby (within 2 meters) and continuously exchange uplink and downlink packets belonging to the VO AC. The second AP

[2]Our study shows that if a station changes its listen interval after association, the per-station data structure maintained by `hostapd` is not updated.

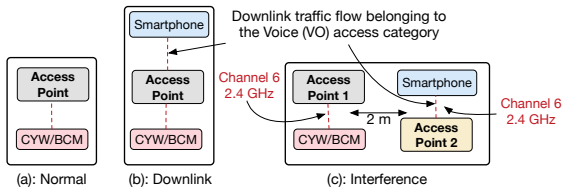


Fig. 3: The scenarios used to evaluate beacon reception. (a): normal, (b) downlink, and (c) interference.

operates on the same channel as the first AP. Figure 3 shows these three scenarios.

The listen interval coefficients used by the IoT station are 1, 5, 10, and 20. During these tests, when a station connects to the AP, it stays connected for 60 seconds, then disconnects and waits for 10 seconds, and then repeats the same process for 2 hours. The same process is followed for all the listen interval values.

B. Results and Discussion

Figure 4 shows the wake-up duration of CYW and BCM per beacon instance. With CYW, increasing the listen interval results in significantly higher wake-up duration (up to 300 ms). Also, this station shows a higher sensitivity to downlink and interference. In contrast, with BCM, increasing listen interval or the presence of downlink and interference results in a slightly (less than 10 ms) higher wake-up duration. We explain these behaviors as follows.

The AP synchronizes stations using the basic Time Synchronization Function (TSF) method by periodically broadcasting the time in beacon frames. The timestamp inside beacons contains the value of the TSF timer when the first bit of the timestamp is handed to the physical layer (PHY) from the MAC-PHY interface, plus the transmission delay from the PHY to the antenna. The receiving STA sets the value of its TSF timer (TSFSTA) to the timestamp present inside the Beacon. In addition to beacons, probe response frames are also used to perform TSF timer synchronization. However, these frames are exchanged only when a station associates with the AP; afterwards, periodic beacons are used by the station to remain synchronized to the AP.

To investigate if CYW can wake up on time to receive beacons, we measure the actual interval between beacon receptions for both stations. Our results show that CYW's beacon loss rate increases versus listen interval because it cannot properly adjust its timer to wake up on time, even in the absence of downlink and interference. By having a closer look into the wake-up duration before each beacon reception, we noticed that this duration almost doubles for each step of increasing listen interval coefficient (we do not include the results due to space limitation); thereby confirming the impact of time synchronization on wake-up duration.

The presence of downlink traffic or interference causes further increase in wake-up duration. With downlink traffic, which belongs to the VO AC in these experiments, beacon

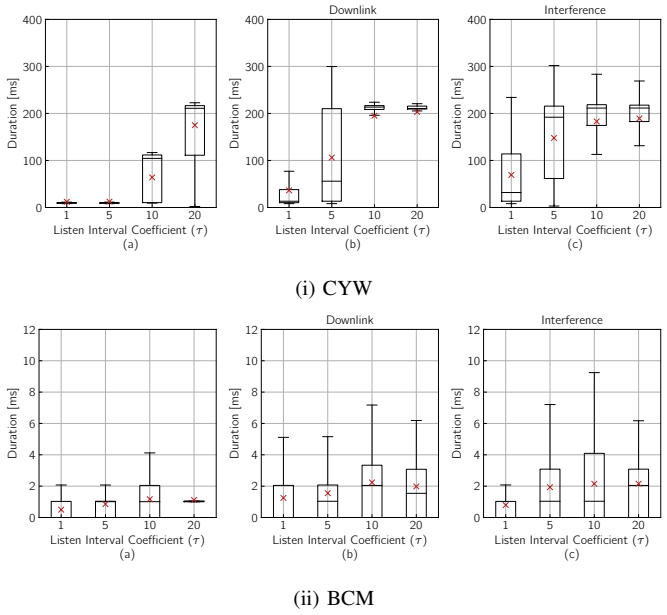


Fig. 4: Per-beacon wake-up duration. Cross marks represent the mean values. These results show that the duration of receiving each beacon is highly affected by increasing listen interval, concurrent traffic, and interference when using the CYW station. This increase is marginal for the BCM station.

packets are delayed by the Transmit Opportunity (TXOP) of VO AC. This is due to two reasons: First, the VO AC defines a 1.504 ms TXOP period during which no other transmission can take place. This is demonstrated in Figure 5. Second, the default AC of beacon packets is BK, which has a lower priority compared to VO. With interference, beacon packets are subject to collision and loss.

To maintain its association with the AP, the station needs to receive a certain number of beacons per second, depending on the driver implementation. When the beacon interval is 1, even if the station does not receive a beacon (due to delay or loss) at some beacon instances, the station can simply switch back to sleep mode and look for beacon during the next interval. As the listen interval increases, tolerance against beacon loss is reduced due to the need for time synchronization and link quality estimation to the AP. In this case, the station stays awake for a longer duration to look for incoming packets.

The main takeaway is that *increasing listen interval cannot be simply used as a method to improve energy efficiency because transceivers show different overheads considering listen interval duration, beacon transmission delay, and interference.*

As the value of τ increases, the chance of receiving multicast and broadcast packets is dropped. This is because the DTIM value on commercial APs is usually between 1 to 3, which means multicast and broadcast packets are transmitted every 1 to 3 beacon intervals. Various solutions can be used to address this problem when $\tau > 1$. For example, the AP may convert multicast and broadcast packets into unicast packets. For ARP packets, the AP may implement a proxy and reply to queries without having to involve stations.

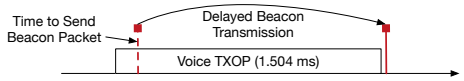


Fig. 5: The presence of a flow belonging to the Voice (VO) Access Category (AC) can result in delayed beacon transmission.

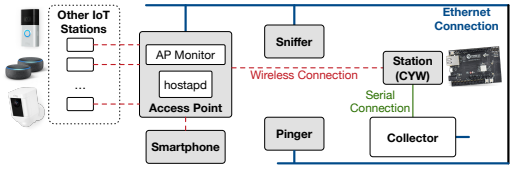
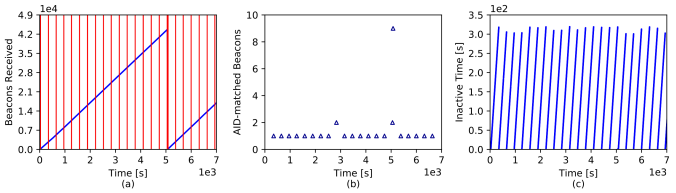
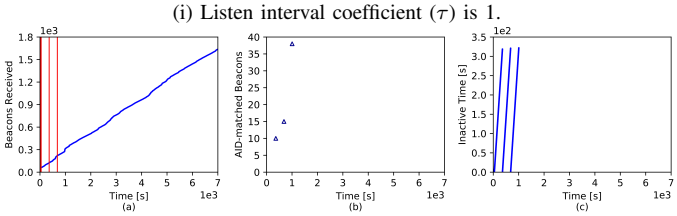


Fig. 6: The testbed used for studying parameters affecting association durability.



(i) Listen interval coefficient (τ) is 1.



(ii) Listen interval coefficient (τ) is 20.

Fig. 7: AP-based polling and the impact of listen interval. In sub-figures (i) and (ii): (a) Number of beacons received by the station (blue lines) as well as the communications between the AP and station (red bars). (b) Number of AID-matched beacons (indicating buffered packets for the station) per 30 seconds intervals. (c) Inactivity time of the station from the AP point of view.

VI. Maintaining Association

Associated stations need to periodically communicate with the AP to renew their *inactivity timer* maintained by the AP. Referring to `hostapd` configuration, if a station does not communicate with the AP for `ap_max_inactivity` duration, the AP either disassociates and deauthenticates the station immediately, or sends a poll frame to the station and maintains the association if a response is received. This configuration is available via the `skip_inactivity_poll` flag. Maintaining association is particularly important to ensure: (i) IoT stations spending most of their time in sleep mode can communicate with the AP without having to reassociate, and (ii) the AP can deliver downlink packets to the station.

In this section, we identify the challenges and present solutions to maintain association.

A. Configuration

Figure 6 shows the testbed used for the studies of this section. AP Monitor is a user-space daemon running on the AP to monitor association status, connected time, and idle time of the station. This daemon continuously transmits its status to the Collector via an Ethernet connection. Sniffer is used to capture the packets exchanged between the AP and the station. The sniffer reports to the collector via an Ethernet connection. Collector is connected to the station via a serial connection cable to collect the number of received beacons by the station during the current association period. Pinger is used to ping the station and measure RTT. The AP, Sniffer, and Pinger report their statistical information to the Collector via Ethernet.

This testbed was set up in a home environment where, in addition to the IoT station under test (referred to as *station* above), there are 12 other IoT stations (such as cameras, thermostat, and Amazon Echos/Dots) associated with the AP. The AP is placed in the family room and the IoT station is placed in a bedroom. The AP-station distance is around 14 meters, and the RSSI perceived by the station varies between -40 to -55 dBm.

B. Maintaining Association by Probing

In the first experiment, we evaluate the effect of listen interval on connectivity. Since modifying the duration of `ap_max_inactivity` on commercial APs is not possible, we use its default value of 300 seconds in our experiments. The `skip_inactivity_poll` is disabled. Figures 7(i) and (ii) show the results when using $\tau = 1$ and $\tau = 20$, respectively.

In Figure 7(i), the AP can successfully poll the station every 300 seconds until 5000 seconds. At this point, as we can observe in Figure 7(i)(b), the number of beacon packets that include the AID of the station is suddenly increased to 9. A closer look revealed an interesting behavior. After the first beacon packet, the station needs to send two PS-Poll packets to receive an ACK from the AP. The AP then sends 6 Null packets to the station, but no ACK is received. Then, collision avoidance mechanism (RTS/CTS) is used to communicate with the station, and 20 RTS packets are sent. Meanwhile, `hostapd`'s probing timer expires and the station is disassociated by the AP. Fortunately, link reliability improves shortly and the disassociation packet is conveyed to the station after the next beacon instance. This scenario clearly shows the impact of link unreliability on disassociation. Link unreliability is caused by various factors, including interference and the distance between two nodes. The main takeaway is that *even when the listen interval matches the beacon period, the internal timer of* `hostapd` *prevents the AP from making enough attempts to poll the station and maintain association.*

Figure 7(ii) presents the same experiment with $\tau = 20$. Two polling events at 300 and 600 seconds are performed successfully. The third polling, however, fails because the AP does not hear back PS-Poll from the station. We observed that AP sends 35 beacons that include the station's AID. In this case, the AP assumes that the station is no longer in range; thereby, it does not send a disassociation packet to the station. Therefore, as Figure 7(ii)(a) shows the trend of receiving beacons from the AP, *from the station's point of view, the station is still connected to the AP.* A *disassociation-unaware station* does not make any attempt to re-associate

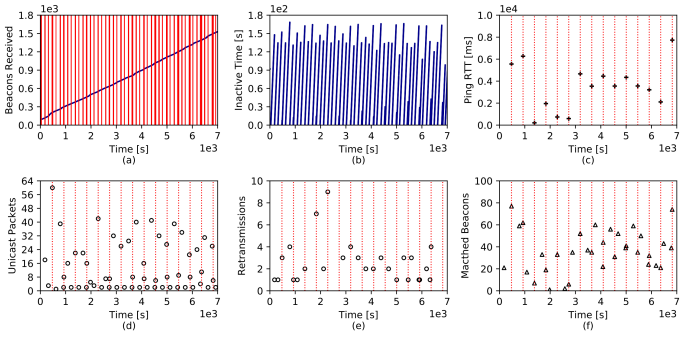


Fig. 8: Maintaining association by using station-side keep-alive packets. Listen interval coefficient (τ) is 20. (a) Number of beacons received by the station (blue line) as well as the communication instances between the AP and station (red bars). (b) Inactivity time of the station from the AP point of view. (c) RTT of pinging. (d) Number of unicast packets exchanged between the station and AP per 30-second intervals. (e) Number of retransmissions between the station and AP per 30-second intervals. (f) Number of AID-matched beacons sent by the AP per 30-second intervals. The dotted lines in (c), (d), (e), and (f) indicate ping transmission instances.

with the AP as long as it receives the beacon packets. For event-based applications that trigger uplink traffic (e.g., video streaming when motion is detected, reporting temperature when a threshold is passed), this behavior may require the station to re-associate whenever inter-event interval is longer than 300 seconds. For applications that require downlink communication with stations, the disassociation from the AP prevents the user or cloud platform from communicating with the station. The main takeaway is that *maintaining association by relying on the AP to send probes is unreliable because the station may be left unaware of disassociation event.*

C. Maintaining Association by using Keep-Alive Packets

The above problem with AP-side probing can be eliminated by using station-generated keep-alive messages. A keep-alive message can be generated by sending a UDP, TCP, Null or ARP packet. Some low-power transceivers allow keep-alive packet generation without the involvement of application processor. Both CYW and BCM support this feature. To validate the effectiveness of transceiver-generated keep-alive packets, we utilized driver APIs to set up Null packet generation. Also, to verify successful downlink packet delivery, the Pinger (see Figure 6) pings the station every 450 seconds.

Figure 8 shows the results for $\tau = 20$. In terms of connectivity, Figures 8(a) and (b) show that the association is maintained by both parties. The inactivity timer kept by the AP for the station is renewed every 150 seconds, which corresponds to a Null keep-alive packet or the packet exchange caused by pinging. Figure 8(c) confirms that ping message is always delivered to the station, although some instances are as long as 8 seconds. Comparing Figures 8(c) and (f) reveals the relationship between the number of AID-matched beacons and ping delay. Specifically, the main cause of communication delay is the number of beacon intervals that the AP needs to send AID-matched beacons until a successful packet exchange.

To better understand the impact of link unreliability, assume that the probability of packet arrival (ping in this case) at the AP between each wake-up instance of a station is a uniform distribution. In the absence of packet loss, for a listen interval coefficient τ, the average number of times the AP needs to send AID-matched beacons before receiving the PS-Poll packet is $(\tau - 1)/2$. For this experiment, this theoretical average is 9.5. However, our results show an average value of 36, which is caused by packet loss. The impact of packet loss can be observed in Figures 8(d) and (e). Figure 8(c) reflects the high number of packet exchanges required for the AP and station to communicate. Figure 8(d) shows that packet retransmission was necessary for both pinging and keep-alive delivery.

D. Key Renewal

In the previous sections, we showed the importance of periodic communication with AP to renew its inactivity timer. In this section, we confirm that key renewal is also a time-critical operation and may result in disassociation.

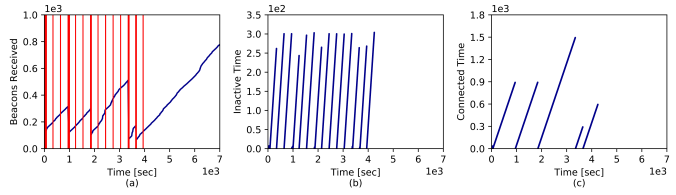
With RSN, each station is assigned two keys: a Pairwise Transition Keys (PTK) for unicast communication, and Group Temporal Key (GTK) for multicast and broadcast communication. Both keys are periodically renewed based on the timer values defined in `hostapd`'s configuration. The GTK is also renewed whenever a station leaves the network. This renewal is necessary to prevent the station from receiving multicast or broadcast packets of the network that the station no longer belongs to. This means mobility and disassociation of any station may require the AP to communicate with all stations.

In the experiments of this section we use a smartphone (see Figure 6) to trigger GTK renewal. The smartphone has been programmed to leave the network every 450 seconds and then join again after 3 seconds. Figure 9(i) shows the results with the default configuration of `hostapd`. As it can be observed, the GTK renewal is performed successfully until around 4000 seconds. At this time, due to link unreliability, the AP fails to renew the key and disassociates the station. However, since the station does not receive the disassociation packet, the result is a *disassociation-unaware station*.

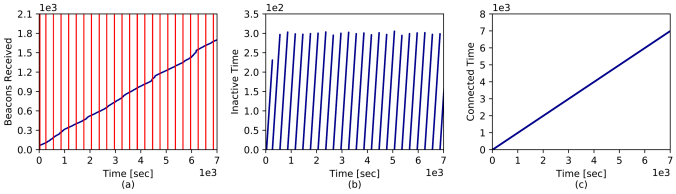
Although the above behavior is observed with the default configuration (commercial APs), `hostapd`'s configuration allows us to increase the number of key renewal retries. To leverage this feature, we increase the value of `wpa_group_update_count` to 32, compared to its default value of 4. Figure 9(ii) confirms that with the new value, the AP can successfully update the key, even in the presence of long listen interval and link unreliability. The main takeaway is that *key renewal may result in the disassociation of IoT stations when the listen interval is long or when the communication link is unreliable. And fixing this problem requires customizing the configuration values of* `hostapd`.

VII. Related Work

Abedi et al. [4] argue that the physical layer energy consumption range of 802.11 and BLE are 10-100 nJ/bit and 275-300 nJ/bit, respectively. They also discuss that the reason



(i) `wpa_group_update_count=4`



(ii) `wpa_group_update_count=32`

Fig. 9: Impact of key renewal on association durability. Listen interval coefficient (τ) is 20. In both sub-figures (i) and (ii): (a) Number of beacons received by the station (blue lines) as well as the communications between the AP and station (red bars). (b) Inactivity time of the station from the AP point of view. (c) Connected time of the station from the AP point of view.

the overall energy consumption of 802.11 is higher than that of BLE is due to the cost of association and maintaining association. They propose Wi-LE, which eliminates these costs by placing data inside beacon packets. Seneviratne et al. [9] show that the loss of DHCP packets is the main cause of association failure and delay. Pei et al. [8] conducted a large-scale empirical measurement and revealed that up to 45% of the users experience association failure, and 15% of successful associations take more than 5 seconds. They also showed that the scan process contributes to 47% of association delay. Tozlu et al. [13] evaluated the impact of various configurations on the energy efficiency of 802.11 stations. Their study shows that WPA2/AES-PSK achieves the highest security-performance tradeoff. Also, using keep-alive packets is more efficient than re-association every one hour. Chen and Qiao [14] propose a handoff mechanism that does not require the stations to dwell on each scanned channel to receive the response messages. Once a station sends a probe message, it switches back to its current channel and delegates the task of collecting information from nearby APs to the currently associated AP.

Vinhas et al. [6] show that the energy consumption of PSM is 4x and 2.5x higher than that of APSD when the delay between wake-up periods is 40 ms and 2 ms, respectively. Chen et al. [15] propose M-PSM for scenarios where the load of stations is light and delay-insensitive. The ultimate goal of M-PSM is to retrieve the packets of stations when the link quality to the AP enables the adoption of higher link rates. A per-station DTIM allocation mechanism has been proposed in [7]. Instead of maintaining one DTIM for all the stations, the AP enables the stations to request their own desired DTIM period. A scheduling mechanism for reducing energy waste due to idle time has been proposed in [2].

None of the existing works study the impact of listen interval, key offloading during association, operating system, network stack, AP inactivity timer, and key renewal on energy efficiency.

VIII. Conclusion

In this paper, we performed an empirical evaluation of the three fundamental operations—association, beacon reception, and maintaining association—that their energy efficiency is essential to build low-power 802.11-based IoT systems. In summary, our results substantiate that: (i) association cost depends on multiple factors including probing, key generation, operating system, and network stack, (ii) increasing listen interval to reduce beacon reception overhead may negatively impact energy efficiency, (iii) maintaining association by relying on AP poll messages is not reliable, and (iv) key renewal aggravates the chance of disassociation. We also presented station- and AP-side solutions to address these problems. In addition to identifying the challenges of building low-power IoT systems with commercial APs, this research also highlights the importance of firmware customization and transceiver choice based on the application at hand.

Acknowledgment

This project has been partially funded by the Latimer Energy Lab. We would like to thank Infineon Technologies for providing our lab with some of the materials used to conduct this project.

References

[1] C. Gray, R. Ayre, K. Hinton, and L. Campbell, "Smart is not free: Energy consumption of consumer home automation systems," *IEEE Transactions on Consumer Electronics*, 2019.

[2] J. Sheth and B. Dezfouli, "Enhancing the energy-efficiency and timeliness of iot communication in wifi networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9085–9097, 2019.

[3] H. Pirayesh, P. K. Sangdeh, and H. Zeng, "Ee-iot: An energy-efficient iot communication scheme for wlans," in *INFOCOM*, 2019, pp. 361–369.

[4] A. Abedi, O. Abari, and T. Brecht, "Wi-LE: Can WiFi Replace Bluetooth?" in *18th ACM HotNets*, 2019, pp. 117–124.

[5] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *10th ACM MobiSys*. ACM, 2012, pp. 225–238.

[6] A. Vinhas, V. Bernardo, M. Pascoal Curado, and T. Braun, "Performance analysis and comparison between legacy-PSM and U-APSD," 2013.

[7] J. Wang and T. Nagy, "Maintaining delivery traffic indication message (DTIM) periods on a per-wireless client device basis," Patent, Aug. 23, 2011, US Patent 8,005,032.

[8] C. Pei, Z. Wang, Y. Zhao, Z. Wang, Y. Meng, D. Pei, Y. Peng, W. Tang, and X. Qu, "Why it takes so long to connect to a WiFi access point," in *IEEE INFOCOM*, 2017, pp. 1–9.

[9] S. Seneviratne, A. Seneviratne, P. Mohapatra, and P.-U. Tournoux, "Characterizing WiFi connection and its impact on mobile users: practical insights," in *8th ACM WinTech*. ACM, 2013, pp. 81–88.

[10] Cypress Semiconductor. CYW43907: IEEE 802.11 a/b/g/n SoC with an Embedded Applications Processor. [Online]. Available: http://www.cypress.com/file/298236/download

[11] BCM4343W: 802.11b/g/n WLAN, Bluetooth and BLE SoC Module. [Online]. Available: https://products.avnet.com/opasdata/d120001/medias/docus/138/AES-BCM4343W-M1-G_data_sheet_v2_3.pdf

[12] B. Dezfouli, I. Amirtharaj, and C.-C. Li, "EMPIOT: An energy measurement platform for wireless IoT devices," *Journal of Network and Computer Applications*, vol. 121, pp. 135–148, 2018.

[13] S. Tozlu, M. Senel, W. Mao, and A. Keshavarzian, "Wi-Fi enabled sensors for internet of things: A practical approach," *IEEE Communications Magazine*, vol. 50, no. 6, pp. 134–143, 2012.

[14] X. Chen and D. Qiao, "Hand: Fast handoff with null dwell time for ieee 802.11 networks," in *IEEE INFOCOM*, 2010, pp. 1–9.

[15] X. Chen, S. Jin, and D. Qiao, "M-PSM: Mobility-aware power save mode for IEEE 802.11 WLANs," in *31st ICDCS*. IEEE, 2011, pp. 77–86.