Extending Battery Life for Wi-Fi-Based IoT Devices: Modeling, Strategies, and Algorithm

Shyam Krishnan Venkateswaran
shyam1@gatech.edu
Georgia Institute of Technology

Ching-Lun Tai
ctai32@gatech.edu
Georgia Institute of Technology

Yoav Ben-Yehezkel
yoavby@ti.com
Texas Instruments

Yaron Alpert
yaron@ti.com
Texas Instruments

Raghupathy Sivakumar
siva@ece.gatech.edu
Georgia Institute of Technology

ABSTRACT

Wi-Fi is one of the key wireless technologies for the Internet of things (IoT) owing to its ubiquity. Low-power operation of commercial Wi-Fi enabled IoT modules (typically powered by replaceable batteries) is critical in order to achieve a long battery life, while maintaining connectivity, and thereby reduce the cost and frequency of maintenance. In this work, we focus on commonly used sparse periodic uplink traffic scenario in IoT. Through extensive experiments with a state-of-the-art Wi-Fi enabled IoT module (Texas Instruments SimpleLink CC3235SF), we study the performance of the power save mechanism (PSM) in the IEEE 802.11 standard and show that the battery life of the module is limited, while running thin uplink traffic, to ∼ 30% of its battery life on an idle connection, even when utilizing IEEE 802.11 PSM. Focusing on sparse uplink traffic, a prominent traffic scenario for IoT (e.g., periodic measurements, keep-alive mechanisms, etc.), we design a simulation framework for single-user sparse uplink traffic on ns-3, and develop a detailed and platform-agnostic accurate power consumption model within the framework and calibrate it to CC3235SF. Subsequently, we present five potential power optimization strategies (including standard IEEE 802.11 PSM) and analyze, with simulation results, the sensitivity of power consumption to specific network characteristics (e.g., round-trip time (RTT) and relative timing between TCP segment transmissions and beacon receptions) to present key insights. Finally, we propose a standard-compliant client-side cross-layer power saving optimization algorithm that can be implemented on client IoT modules. We show that the proposed optimization algorithm extends battery life by 24%, 26%, and 31% on average for sparse TCP uplink traffic with 5 TCP segments per second for networks with constant RTT values of 25ms, 10ms, and 5ms, respectively.

CCS CONCEPTS

• **Networks** → **Cross-layer protocols**; **Network simulations**; *Network experimentation*; Sensor networks.

MobiWac '21, November 22–26, 2021, Alicante, Spain

ACM ISBN 978-1-4503-9079-8/21/11…$15.00
https://doi.org/10.1145/3479241.3486699

KEYWORDS

Wireless technology; Energy efficiency; IEEE 802.11; Transmission control protocol (TCP); Internet of things (IoT); Sensor network; Network simulation; ns-3; Cross-layer optimization

ACM Reference Format:
Shyam Krishnan Venkateswaran, Ching-Lun Tai, Yoav Ben-Yehezkel, Yaron Alpert, and Raghupathy Sivakumar. 2021. Extending Battery Life for Wi-Fi-Based IoT Devices: Modeling, Strategies, and Algorithm. In *Proceedings of the 19th ACM International Symposium on Mobility Management (MobiWac '21), November 22–26, 2021, Alicante, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3479241.3486699

1 INTRODUCTION

Wi-Fi is fast emerging as one of the prime choices for wireless connectivity for Internet of things (IoT), particularly for indoor and medium-range applications (< 100 meters). Compared to other competing wireless technologies (see [10] for examples), Wi-Fi features advantages of ubiquity and standardization with backward compatibility, and provides toolkits to support system tradeoff between throughput, latency, range, power, and spectral efficiency [9].

A significant proportion of IoT use cases require power saving optimization (i.e., optimal power efficiency), particularly battery-operated sensors with wireless communication capabilities that are deployed over vast areas and hard-to-reach places. However, reducing power consumption in such devices is not straightforward since key connectivity requirements such as wireless range, latency, and throughput depend upon the power drawn.

Power save mechanism (PSM) is the major IEEE 802.11 standard mechanism that enables low-power operation in client devices (explained in Sec. 2.1). Such mechanisms work primarily by allowing coordination between the access point (AP) and the IoT client to power down the client when not in use while the AP buffers incoming frames, making Wi-Fi a viable candidate for low-power IoT. Since PSM is part of the Wi-Fi Alliance (*WFA*) certification, commercial Wi-Fi capable IoT devices often use PSM to conserve power. As will be discussed in Sec. 2.3, there have been works on studying the effect of PSM on power consumption, throughput, packet losses, and latency - all at a high level. However, there is a lack of work that uses extensive experimental analysis of a real-world device to analyze the key components of power consumption in detail. In addition, a majority of the works that aim at increasing the power efficiency of Wi-Fi client devices require modifications to the Wi-Fi standards (e.g., [12]).

The focus of this paper is to study, analyze, and increase the power efficiency of fully standard-compliant low-power Wi-Fi IoT clients. Following are the key contributions:

(1) We conduct extensive experiments with a state-of-the-art Wi-Fi enabled IoT module (Texas Instruments SimpleLink CC3235SF) which relies on the provisions in the IEEE 802.11 standard for power saving, analyze the experimental observations, and identify the components that constitute power consumption to present several key insights. Besides, we show that without PSM, even for a scenario without any data traffic, the battery life of the IoT module is as poor as *two days*, and that with PSM, the battery-life is about *six months*. When a scenario with thin upstream data traffic is considered, the battery life drops down to *two months*.
(2) We design and enhance the ns-3 simulation framework with a detailed and generic power consumption model taking into account implications of sparse traffic and PSM signaling to estimate power consumption considering specific network characteristics (e.g., round-trip time (RTT) and traffic conditions). In addition, we propose five standard-compliant strategies for low-power operation that mitigate the impact of sparse upstream data traffic and calibrate the power consumption model to SimpleLink CC3235SF for simulating these strategies.
(3) With the developed simulation framework, we perform detailed analysis on the implications of specific network characteristics (IEEE 802.11 PSM, network RTT, and frame transmission time with respect to the next beacon reception) on power consumption. In doing so, we establish several practical bounds on power consumption for a common IoT traffic scenario - periodic uplink reporting over transmission control protocol (TCP).
(4) Based on the insights from the analysis, we design a cross-layer power saving optimization algorithm operating within IEEE 802.11 standards by intelligently timing uplink TCP transmissions. With simulations, we show the efficacy of the developed optimization in reducing power consumption.

The rest of the paper is organized as follows: Sec. 2 provides necessary background information and sets up the scope along with problem statement. Sec. 3 presents experimental results and insights. The simulation framework and power consumption model are described in Sec. 4. Sec. 5 describes power saving optimization strategies for low-power operation and simulation results. Sec. 6 presents the optimization scheme, while Sec. 7 concludes the paper and presents future research directions.

2 BACKGROUND AND PROBLEM STATEMENT

2.1 IEEE 802.11 PSM

The IEEE 802.11b specification revision introduced the PSM which allows the client device network interface to power down periodically between beacon receptions. A client device wishing to enter the lower-power *sleep* state informs the AP usually through a null frame with the *Power Management* bit set to 1. Upon receiving acknowledgement (ACK) from the AP, the client enters the sleep state, while the AP begins to buffer frames that are intended for the specific client. The maximum duration for which the AP buffers frames before discarding is not specified in the standard and is up to the AP capabilities and configuration. Typically, APs are required to buffer unicast frames till at least the arrival of the next beacon and multicast/broadcast frames till at least the arrival of the next delivery traffic indication message (DTIM) frame.

The AP informs the specific client of buffered unicast frames via the traffic indication map (TIM) field in beacons. Client devices are typically configured to receive beacons while in the sleep state. If the corresponding association identifier (AID) bit is set as 1 in the beacon, the client wakes up and transmits a *Power-Save Poll* (PS-POLL) frame to the AP to retrieve buffered frames back-to-back till there are no more frames before returning to the sleep state. When using the PS-POLL mechanism to retrieve TCP ACKs, the RTT is rounded up to the next beacon reception (represented as $\lceil RTT \rceil_{BCN}$) [12]. If a client device transmits a PS-POLL frame when there is no buffered traffic, the AP typically transmits back a Null frame with the *More Data* field set to 0. In the presence of more than one buffered unicast frames, the *More Data* flag of the first buffered frame sent from the AP to the client will be set to 1. Note that the client device can wake up and send frames (PS-POLL or others) to the AP at any point for communication within or outside the wireless local area network.

2.2 IoT - Traffic Scenarios

The paradigm of IoT encompasses a wide variety of applications with different traffic profiles. Based on the size of data transmitted within short duration, IoT traffic can be broadly classified into two categories, **bursty traffic** (e.g., surveillance camera) and **lean traffic** (e.g., soil moisture sensor).

In this work, we focus on lean traffic scenarios that generate (uplink) sparse traffic which is either **periodic** or **triggered**. Periodic sparse traffic is very prominent in IoT low-cost sensing use cases that involve periodic, sporadic, or unpredictable reporting of measurements (temperature, moisture, CO_2 levels, etc.) as small packets of data over a period of few seconds. This also occurs in IoT application mechanisms such as uplink TCP for application layer keep-alive packets sent by the IoT client device periodically to remote servers. Unpredictable triggered uplink traffic includes applications such as smoke detectors that generate sparse traffic when triggered by a physical event [13].

2.3 Related Work

The power efficiency of wireless devices is an area of active research, and this section presents a brief review of related works specifically in the context of power consumption modeling and optimization for Wi-Fi IoT devices.

Existing works on power consumption models for wireless devices such as [11], [18] and [7] focus on macroscopic lifetime estimation for wireless IoT devices, simplifying device operation into a limited set of states. In this work, we develop a detailed power consumption model that takes into account state transitions and power operation profiles.

Optimizing radio transmission characteristics is one way for saving power. [8] presents a brief survey of methods using optimal selection of modulation and coding schemes (MCS) which depend on the link quality and traffic requirements. Specifically for wireless sensor networks, [19] proposes an adaptive MCS selection for higher power efficiency. Power efficiency can also be improved by optimizing circuit design, as explored in [2]. Another way to increase battery life is by replenishing batteries through energy harvesting. For instance, authors of [14] improve the power efficiency by developing a novel energy harvesting front-end. Moreover, an interesting approach for low-power wireless connectivity is wake-up radio (WUR) which uses a separate low-power radio for connectivity maintenance [4].

There have also been works on analyzing and enhancing the PSM in IEEE 802.11 standards. An analysis of PSM is presented in [3], exploring the effectiveness of PSM for Wi-Fi hotspots in terms of packet loss probability, RTT, and the number of users. A similar study of PSM on the application throughput, packet losses, and power consumption at the client devices is presented in [17]. In [12], an optimal algorithm called Bounded Slowdown is developed, which minimizes power consumption while guaranteeing that the latency will be bounded. In [15], the authors propose PSM with adaptive wake-up (PSM-AW) to maximize the sleep time period of client devices while maintaining acceptable performance.

Note that the above presented works, in most cases, require modification of Wi-Fi standards or adoption of the WFA certification. In this work, we propose a client-side optimization that is standard-compliant and can work with any WFA-compliant AP.

2.4 Scope and Problem Statement

The focus of this paper is the power consumption of Wi-Fi enabled IoT client devices that can operate with commodity (consumer) WFA certified Wi-Fi APs. Currently, state-of-the-art Wi-Fi enabled low-power IoT modules are capable of operating within the IEEE 802.11 b/g/n standards and rely on IEEE 802.11 PSM to reduce power consumption. In addition, we focus on single-user scenarios where power consumption optimization is achieved for client devices independently of the existence and capabilities of other clients, considering the power efficiency performance of client-side optimization which can fully interoperate with any WFA certified AP without modifications to IEEE 802.11 standards. Besides, only power saving optimization strategies that do not compromise application throughput and latency requirements are considered. For example, beacon skipping, a technique where the client device chooses to skip receiving beacons for periodic intervals at the cost of increasing application latency, is beyond the scope. For the same reason, data aggregation techniques where IoT data are buffered at the client while the client is in low-power states are not considered. Similarly, optimizing the DTIM period to enable longer sleep periods is not explored as it requires changes to AP configuration.

In this paper, we study the sparse uplink IoT traffic described in Sec. 2.2 since it represents a prominent IoT use case, and address the problem of reducing power consumption of Wi-Fi enabled IoT clients (thereby increasing their battery life) with sparse uplink traffic profiles through client-side optimization that is compliant with IEEE 802.11 standards.

3 EXPERIMENTAL ANALYSIS OF A STATE-OF-THE-ART IOT MODULE

The state-of-the-art Wi-Fi enabled low-power IoT module *SimpleLink CC3235SF* [1] from Texas Instruments is used in this paper for experimental analysis and for calibration of the developed power consumption model.

3.1 Texas Instruments CC3235SF

The *IoT module* supports IEEE 802.11 a/b/g/n dual-band (2.4 and 5 GHz) Wi-Fi with an on-board application microcontroller (32-bit Arm Cortex-M4) which is capable of running user applications. This system-on-chip (SoC) architecture is optimized for low-power wireless operation and supports wireless security features up to and including Wi-Fi protected access 3 (WPA3). The battery-powered module allows user applications to configure and utilize built-in low-power profiles depending on the application requirements.

3.2 System Setup

3.2.1 Components of the Experiment

The experimental setup in Fig. 1 consists of five major components: Wi-Fi IoT module (explained in Sec. 3.1), current measurement module, Wi-Fi AP, TCP/UDP server (laptop), and network sniffer.
Current measurement module: The *IMETER-BOOST* BoosterPack[2] housing a high-accuracy power monitor *INA226* is used for current measurement in this work with an accuracy of 5 μA at a minimum sampling interval of 0.1 μs. IMETER-BOOST is connected in series with CC3235SF using jumper wires for measurement.

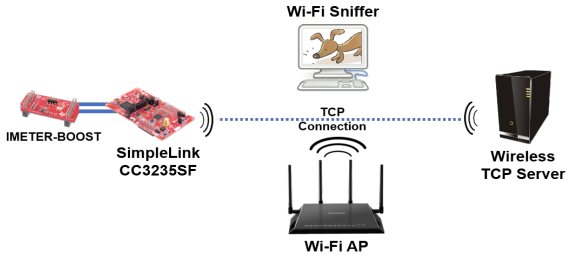


Figure 1: Experimental setup

Wi-Fi AP: Netgear R7500[3] is a high-performance wireless router supporting IEEE 802.11 a/b/g/n/ac in the 2.4 and 5 GHz bands that is used as the Wi-Fi AP (*AP*).
TCP server: An Acer TravelMate P648 laptop running Ubuntu Linux equipped with Qualcomm Atheros QCA6174 Wi-Fi card supporting IEEE 802.11 a/b/g/n/ac is used as the TCP/UDP server (*Server*).
Wi-Fi sniffer: A dedicated Ubuntu machine equipped with Alfa AWUS036ACH USB Wi-Fi adapter running Wireshark in monitor mode is used as the network sniffer (*Sniffer*).

3.2.2 Experimental Conditions and Metrics

The IoT module, AP, Server, and Sniffer are in the same room within 10 meters of each other and with line-of-sight (LoS) connectivity with each other. The IoT module and Server are the only devices connected to the Wi-Fi network hosted by the AP over the 2.4 GHz

[1] https://www.ti.com/product/CC3235SF
[2] https://www.ti.com/tool/IMETER-BOOST
[3] https://www.netgear.com/support/product/R7500.aspx

band. The network RTT is estimated to be 4.1 *ms* on average. The AP is configured with the DTIM period of 1 and beacon period (T_{BCN}) 102.4 *ms*. The experiments are carried out in an academic building with multiple active Wi-Fi networks available. While the IoT module is programmed to a specific power configuration for specific experiments, the MCS and PHY type are chosen adaptively based on the network conditions and traffic requirements.

For evaluation, we assume that the IoT module is powered by a 3V battery of capacity (C_B) 3000 mAh hereafter. The average current (I_{avg}) is measured over an interval of 1024 ms (ten beacon periods) unless mentioned otherwise. The expected battery life (T_{EB}) is calculated in hours (h) as

$$T_{EB}(h) = C_B(mAh)/I_{avg}(mA).$$

Another metric used is the effective network RTT (RTT_{eff}) as seen by the IoT application. For uplink TCP traffic, we define RTT_{eff} as the time interval between the transmission of a TCP segment and the reception of the corresponding TCP ACK, taking into account the effects of the evaluated scheme.

3.3 Experimental Results

3.3.1 Default Power States and Profiles

The IoT client can switch to and remain in low power modes to save energy in appropriate scenarios. The availability of multiple power states is common in low-power IoT clients to tailor for different applications. Specifically, the module can be in one of the following power states:

Shutdown is the lowest-power state. The module is powered off and memories are not retained. The average current is negligible and measured to be less than 5 μA (1 μA from datasheet).

Hibernate is the lowest-power state keeping the real time clock (*RTC*) running. Device is powered off except for the hibernate logic and memories are not retained. The module deauthenticates from the associated Wi-Fi network before entering this state, and hence must initiate authentication and association upon wake-up. The average current is measured to be less than 5 μA (4.5 μA from datasheet).

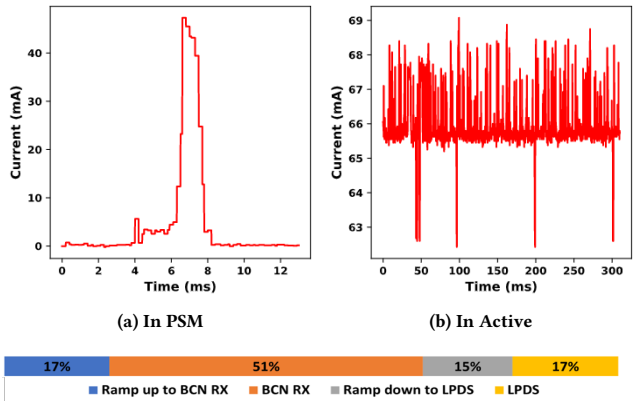
Low-power deep sleep (LPDS) is a sleep state where device voltage levels are lowered and fast clocks are off. Memories are in retention mode (read/write not possible). The module consumes an average current of 120 μA while in this state.

Active is the operational state where both the application microcontroller unit (MCU) and the network processor (NWP) are on with the module always listening. When there is no active transmission, the module consumes 66 *mA* on average in this state.

It is seen that, without any network activity, the battery discharges in less than two days with the *Active* state while lasts for over 1000 days with the LPDS state.

3.3.2 Beacon Reception

When the module is configured in PSM, it switches to the LPDS state between beacon receptions as shown in Fig. 2a, consuming an average current of 0.67 *mA*. For comparison, Fig. 2b shows that the module consumes an average current of ~66 *mA* when receiving beacons without any transmission in the Active state. When configured to only receive beacons with no traffic, the battery will last



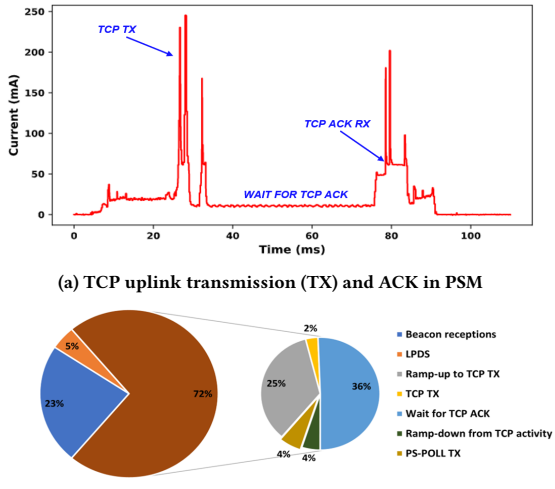
(a) In PSM (b) In Active

(c) Power consumption percentage in PSM over one beacon period

Figure 2: Beacon reception with CC3235SF

for 2 days and 6 months in Active and PSM profiles, respectively, highlighting the significance of PSM. According to the frame capture from the sniffer shown in Table 1, the duration of the beacon frame (including the preamble) is 2120 μs. However, as shown in Fig. 2a, the average duration of the module not being in the LPDS state is 4600 μs. This is because the module spends time waking up from and going back to the LPDS state before and after beacon reception. These *ramp-ups* and *ramp-downs* happen when the module transitions between any two states and must be accounted for while estimating power consumption. In Fig. 2c, we observe that 32% and 51% of the energy are spent in state transitions and beacon reception, respectively.

3.3.3 Power Usage with TCP Uplink Traffic



(a) TCP uplink transmission (TX) and ACK in PSM

(b) Power consumption percentage in PSM over one second

Figure 3: Power usage with TCP uplink traffic

TCP is generally the chosen protocol for reliable communication over the Internet. We consider periodic uplink sparse traffic (as described in Sec. 2.2) wherein one or more segments are sent every specified time interval. The module is configured to send one TCP segment with 1460 bytes of data every second to the TCP server

while taking advantage of IEEE 802.11 PSM (described in Sec. 2.1). Fig. 3a shows the current profile of data transmission and ACK reception with regard to 1 TCP segment. We observe the rounding up effect of PSM on RTT as RTT_{eff} is rounded up to the next beacon reception [12]. The average current is measured as 2.26 mA corresponding to a battery life of 1.84 (~2) months.

The major components of power consumption (including subcomponents associated with the component of TCP activity) are shown in Fig. 3b. TCP ACK reception and IEEE 802.11 ACK RX/TX consume less than 1% of power and hence are not shown. We observe that 72% of power is consumed by TCP activity and 23% by beacon receptions. Within the 72% of power consumed by TCP activity, waiting for TCP ACKs accounts for 36%, while ramp-ups to and ramp-downs from TCP activity accounts for 29%.

Key Insights: IEEE 802.11 PSM is very effective in reducing power consumption (e.g., reducing the average current for beacon receptions by ~99%). We also notice the significance of transitional power states in device power consumption with over 32% contribution during beacon reception and 29% contribution during TCP activity. With sparse uplink traffic at the rate of 1 TCP segment per second, the battery life reduces from 6 months (without traffic) to 2 months, which shows the significance of data traffic on battery life. Even though PSM drastically reduces power consumption, we note its adverse effect on RTT_{eff}, which increases ($\sim \lceil RTT \rceil_{BCN}$) and therefore increases the duration over which the buffer for unacknowledged TCP segments needs to be powered.

4 A NOVEL SIMULATION FRAMEWORK FOR POWER CONSUMPTION MODELING

To analyze the effect of network conditions, device behavior, and other factors on power consumption and investigate alternative power saving optimization strategies, we develop[4] a generic but comprehensive power consumption model to estimate the power consumption of IoT client devices under sparse uplink TCP traffic. The power consumption model is designed to function on top of Network Simulator 3 (ns-3) and is then calibrated to accurately estimate the power consumption characteristics of SimpleLink CC3235SF. In this section, we present the simulation platform and the power consumption model, outline five power saving optimization strategies for device operation along with simulation results, and finally summarize several key insights.

4.1 ns-3 Setup

ns-3 is an open-source discrete-event modular network simulator ([6], [16]) that provides models of data networks along with a versatile simulation engine. The availability of models for IEEE 802.11 standards and the capability to simulate networks with a high degree of control make ns-3 an ideal choice for simulation. We begin by emulating the experimental setup as a topology as shown in Fig. 4 where the configurable parameters for simulation trials are in the boxes below network components. There are three stationary network nodes: IoT client node **(STA)**, Wi-Fi AP, and TCP server **(Server)**, which model their namesakes respectively. The STA is served by the Wi-Fi network created by the AP with LoS connectivity. Each frame (relevant to the purposes of the power consumption model) originating from the AP and the STA are matched in specification for the experiments, as shown in Table 1.

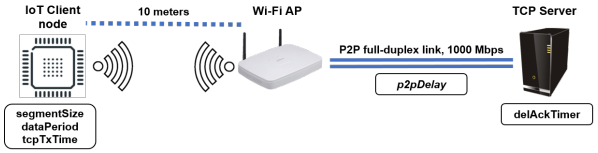


Figure 4: Network topology and configurable parameters

The AP and Server are connected via a wired full-duplex link using the point-to-point (P2P) model with a high capacity (1000 Mbps) such that the link does not risk becoming a bottleneck. The simulation is set up with the following configurable parameters: TCP segment size (***segmentSize***) and data period (***dataPeriod***) to characterize the uplink traffic profile, time instant of TCP segment transmission (***tcpTxTime***), link delay of the P2P link (***p2pdelay***) to vary the network RTT, and TCP delayed ACK timer (***delACKTimer***) at the TCP server. We utilize two trace files to create the power consumption model:

ASCII Trace is a plain text file consisting of all network events with timestamps, frame specification, and protocol information.

PHY state log of STA contains information about the STA's PHY state at every time instant, including Transmit (Tx), Receive (Rx), and IDLE states.

Although ns-3 provides an energy framework to model power consumption [18] that is compatible with its Wi-Fi PHY model, there are three major limitations related to sparse traffic power consumption modeling:

Absence of transitional power states: A key inference from experiments is the significance of transitional power states (e.g., switching between OFF, RX, TX, and IDLE) in terms of power consumption. Switching between states involves multiple-operation transitional mechanisms such as stabilization/gating of clocks and (de-)activation of analog components and hence consumes time and energy. Therefore, transitional power states must be accounted for in the power consumption model. Note that for sparse traffic the impact of transition between states on power consumption is significant and should not be neglected.

Limited choice of system power states: Multiple low-power states and the ability to switch between these states are common in very low-power IoT client devices (such as *SLEEP* and *SLEEP_BUFFER* states from Table 2). In addition, IEEE 802.11 PSM is not implemented as a part of the Wi-Fi device and standard models. The power consumption model should have provisions to define system power states and to implement and model IEEE 802.11 compliant power saving mechanisms.

Traffic content unawareness: Power consumption for RX and TX states may depend upon the type of traffic. For example, beacon reception may consume less power than TCP segment reception. Traffic content should be factored in by the power consumption model for defining current consumption values per component while performing a specific operation flow. Due to the inherent sparse nature of IoT traffic, the entropy of packet types increases dramatically.

[4] Available at https://github.com/shyam100v/LowPowerWiFi-IoT

Packet/Frame	Specification	Experiment	Simulation	Communication direction
Beacon	PHY type (rate)	802.11b (HR/DSSS)		AP $\xrightarrow{Broadcast}$ STA
	Size (duration)	217 bytes (1928 μs)	147 bytes (1368 μs)	
	Preamble duration	192 μs		
TCP Data	PHY type (rate)	802.11n (72.2 Mbps)		STA $\longrightarrow$ Server
	Size (duration)	1554 bytes (209 μs)	1550 bytes (209 μs)	
	Preamble duration	36 μs		
TCP ACK	PHY type (rate)	802.11n (52 Mbps)	802.11n (57.8 Mbps)	AP $\xrightarrow{Forward}$ STA
	Size (duration)	94 bytes (52 μs)	90 bytes (50 μs)	
	Preamble duration	36 μs		
IEEE 802.11 ACK	PHY type (rate)	802.11g (24 Mbps)		AP $\longleftrightarrow$ STA
	Size (duration)	14 bytes (28 μs)		
	Preamble duration	20 μs		

Table 1: Packet/Frame specifications

4.2 Power Consumption Model

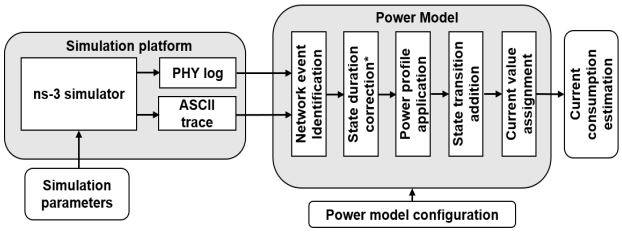


Figure 5: Simulation platform and the IoT power consumption model

We design a power consumption model to mitigate the above-mentioned limitations and to achieve realistic power consumption assessment for the *sparse IoT traffic scenario*. The IoT traffic power consumption model is developed as a wrapper script (written in Python) running on top of the ns-3 simulator and processes, taking the two trace files mentioned in Sec. 4.1 according to a specified configuration and outputting the current consumption, based on ns-3 events, as a time series with the sample interval of 1 μs.

The work flow of the IoT traffic power consumption model and the simulation platform is shown in Fig. 5 and involves the steps described below.

Network Event Identification: The PHY state log is used to identify the start time and the duration of all Tx and Rx states during the time interval of interest. This information is correlated with the ASCII trace, and the network event corresponding to each Tx and Rx state is identified. For example, if an Rx state at a specific time instant corresponds to a beacon frame, then the time instant (and the corresponding duration) is tagged as *BCN_RX*.

State duration correction: In Table 1, the beacon duration in experiments is different from that in simulation due to a difference in the beacon size. To accurately calibrate the power consumption model for a fair comparison with experimental results of a Wi-Fi device, the duration of frame transmissions and receptions must match between experiments and simulations. Thus, to enable calibration, the model allows to selectively change the duration of states by a constant factor. In this case, all *BCN_RX* (beacon reception) states are inflated by a factor of 1.41 ($\approx 1928/1368$).

Power profile application: The power consumption model supports modeling and analysis of IEEE 802.11 standard-based power save signaling and state transition power profiles (including PSM; see Sec. 5.1) with provisions to activate and de-activate power saving profiles conditionally. The standard power save profile (PSM) is also provided as an input to the model with which sate manipulations will be carried out. This includes:

(1) Introduction of low-power sleep mode by replacing *IDLE* states between *BCN_RX* states with *SLEEP* states.
(2) Emulation of IEEE 802.11 PSM by manipulating the time instant of occurrence of TCP acknowledgement reception (*TCP_ACK_RX*) state to be after the next *BCN_RX* state. The model accounts for average delays and the power consumption impact of PS-POLL signaling mechanism.

State transition addition: With every steady power state identified based on the network events, traffic type, and the power profile, this step accounts for the power implications of state transitions. For any pair of states $S1$ and $S2$, the transitions are defined using four parameters: $A_{S1\rightarrow S2}$, $D_{S1\rightarrow S2}$, $A_{S2\rightarrow S1}$, and $D_{S2\rightarrow S1}$, where $A_{S1\rightarrow S2}$ and $D_{S1\rightarrow S2}$ are the average current and duration of the transition $S1 \rightarrow S2$ (similarly for $S2 \rightarrow S1$). **Current value assignment**: With every state and state transition accounted for, the IoT traffic power consumption model uses the sequence of states and their corresponding duratios along with a current value to output a time series of current values which can then be visualized and used for further analysis.

In this paper, we focus on the optimization for continuous IoT operation scenarios, considering only the time intervals within which there is ongoing sparse uplink TCP traffic to explore the power efficiency implications of TCP communication. This means that the STA has already successfully associated with the AP and has established a TCP connection with the server.

To align the power consumption model with a specific IoT client hardware and software implementation, a calibration should be conducted. Our power consumption model was calibrated (and then validated) to match SimpleLink CC3235SF. The calibration for the IoT traffic power consumption model was conducted by estimating the average current of each state and the five parameters (segmentSize, dataPeriod, tcpTxTime, p2pdelay, and delACKTimer) described in Sec. 4.1 for state transitions. As part of calibration, a subset of calibrated parameters for SimpleLink CC3235SF are

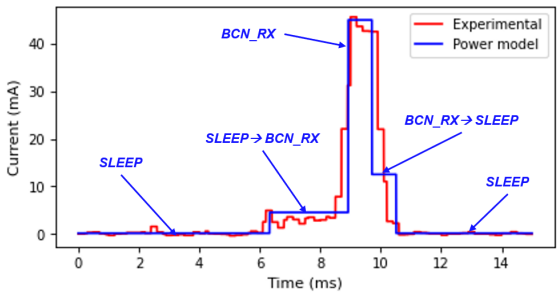


Figure 6: Beacon reception - model vs. experimental

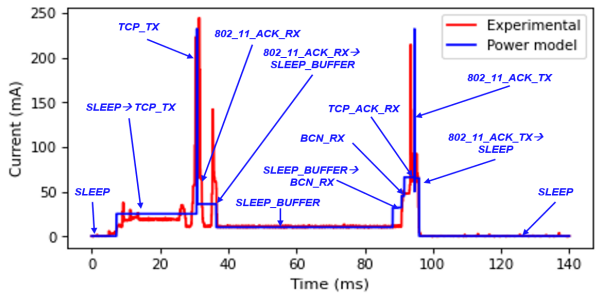


Figure 7: TCP uplink traffic - model vs. experimental

shown in Tables 2 and 3. Note that SLEEP_BUFFER is the power state when the module is waiting for beacons with unacknowledged TCP segments in low power sleep.

Fig. 6 shows the current consumption of beacon reception from both experimental results and the power consumption model calibrated to within 1% of error on average (compared to experimental results), and highlights each state and state transition. Note that calibration is carried out in a piece-wise fashion by matching frame specifications and inflating beacon duration as discussed.

For the case of TCP uplink traffic, the comparison between experimental results and the calibrated power consumption model is shown in Fig. 7 with the states (considered in the power consumption model) labelled. In case of PSM, the power consumed for the PS-POLL mechanism is accounted for in the state transition leading up to TCP ACK reception. Similar to beacon reception, the power consumption model is calibrated to within 1% of error.

State	Average Current
SLEEP	0.12 *mA*
ACTIVE	66 *mA*
BCN_RX	45 *mA*
TCP_TX	232 *mA*
ACK_802_11_RX	50 *mA*
SLEEP_BUFFER	10 *mA*

Table 2: Power profile configuration - steady states

State transition	Average Current	Duration
SLEEP→ BCN_RX	4.5 *mA*	2.6 *ms*
BCN_RX→ SLEEP	12.5 *mA*	0.8 *ms*
SLEEP→ TCP_TX	25 *mA*	23.5 *ms*
TCP_TX→ SLEEP_BUFFER	36 *mA*	5.5 *ms*

Table 3: Power profile configuration - state transitions

5 SIMULATION RESULTS

5.1 Power Saving Optimization Strategies for Sparse TCP Uplink Traffic

We present five IEEE 802.11-compliant client-side power saving optimization strategies (summarized in Table 4 and depicted in Fig. 8) for uplink TCP traffic scenarios. These optimization strategies are generic in the sense that they may be implemented on any IoT module. The underlying motivation is that any specific module would consist of its own set of individual power consumption characteristics/states. A calibrated power consumption model can be used to show which power optimization strategy would be optimal with respect to the specific module's characteristics as well as to the network characteristics. We demonstrate this on the calibrated power consumption model for the CC3235SF module (described in Sec. 4.2).

Power saving optimization strategies: In all five power saving optimization strategies, the client remains in the *SLEEP* state between beacon receptions in the absence of uplink TCP traffic. In strategy (a) **Power Save Mode (PSM)**, TCP ACKs are retrieved through the PS-POLL mechanism and the client enters the *SLEEP_BUFFER* state. In the case of strategy (b) **Long-Term Sleep PSM (LTS-PSM)**, the client enters a lower-power *SLEEP* state while waiting for TCP ACKs. Instead of sending PS-POLL after beacon reception, the client skips the corresponding beacon and transmits PS-POLL at a configured interval after the beacon, taking into consideration the TCP retransmission timeout (*RTO*) and the application latency requirement. The three dynamic PSM strategies (c), (d), and (e) described below involve switching to an Active power profile only during TCP activity and then switching back to PSM while informing the AP accordingly. The difference is in the state that the client enters while waiting for TCP ACKs - *ACTIVE*, *SLEEP_BUFFER*, and *SLEEP* in the cases of **(c) dynamic PSM (dPSM)**, **(d) Low-Power dPSM (LP-dPSM)**, and **(e) LP2-dPSM**, respectively. The calibrated power consumption model is configured to emulate each of the five strategies for analysis. We also analyze the effect of current consumption of specific power states on average power consumption while establishing practical current consumption bounds with regard to PSM.

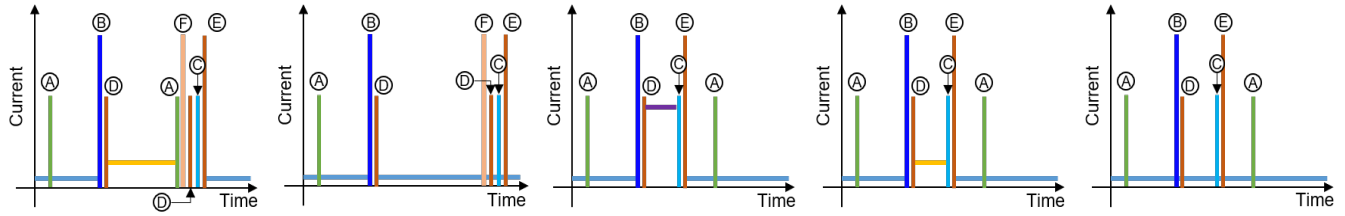
5.2 Simulation and Sensitivity Analysis

The ns-3 simulation setup is configured to simulate and calculate the average current consumed in a period of 1024 *ms* for the five strategies described in Sec. 5.1. Let us define the time interval between the TCP segment transmission and the subsequent beacon reception in each simulation as ***timeToNextBeacon***. The parameter *delAckTimer* is set to 0 *ms*, *segmentSize* to 1460 bytes, and *dataPeriod* to 1024 *ms*. *p2pDelay* is varied in such a way that the resulting network RTT varies from 0.4 *ms* to 200.4 *ms* in steps of 0.5 *ms*. Similarly, *tcpTxTime* is varied between simulations in a manner that *timeToNextBeacon* varies from 1 *ms* to 102 *ms* in steps of 1 *ms*. Totally, it results in over 40,000 sample points for each strategy. It is observed that increasing *delAckTimer* simply translates to an increase in *RTT*. Hence, for the purpose of current estimation, we choose to manipulate *RTT* only through *p2pDelay*.

The resulting samples of average current are plotted as a function of *RTT* and *timeToNextBeacon* in Fig. 9 for each strategy to analyze

	PSM	LTS-PSM	dPSM	LP-dPSM	LP2-dPSM
Power save profile for UL TCP traffic	✓	✓	×	×	×
Power state waiting for TCP ACK	*SLEEP_BUFFER*	*SLEEP*	*ACTIVE*	*SLEEP_BUFFER*	*SLEEP*
RTT_{eff} (latency)	$\sim \lceil RTT \rceil_{BCN}$	$< RTO, > \lceil RTT \rceil_{BCN}$	*RTT*	*RTT*	*RTT*

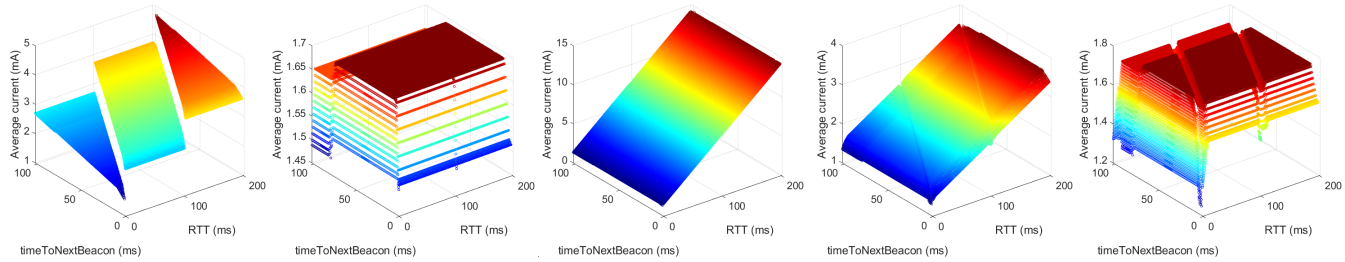
Table 4: Summary of five strategies for IoT TCP uplink traffic



(a) PSM **(b) LTS-PSM** **(c) dPSM** **(d) LP-dPSM** **(e) LP2-dPSM**

Legend: A: Beacon RX, B: TCP TX, C: TCP ACK RX, D: 802.11 ACK RX, E: 802.11 ACK TX, F: PS-POLL TX

Figure 8: IoT TCP uplink traffic with five strategies



(a) PSM **(b) LTS-PSM** **(c) dPSM** **(d) LP-dPSM** **(e) LP2-dPSM**

Figure 9: Average current vs. RTT and timeToNextBeacon

the sensitivity of average current to *RTT* and *timeToNextBeacon*. We note the irregularities in average current for small values of *timeToNextBeacon* and *RTT*, which result from the overlap of state transitions in these cases. Similarly, we observe minor discontinuities in Figs. 9d and 9e at points satisfying eq. (1) below due to overlap of state transitions and beacon receptions. Hereafter, we focus on the general trend of average current consumption with respect to RTT and timeToNextBeacon.

Observations: In case of **PSM**, we observe a jump in average current at points satisfying

$$RTT = (K \times T_{BCN}) + timeToNextBeacon, \quad (1)$$

where $K = 0, 1, 2, \ldots$. At these points, the TCP ACK arrives at the AP just after a beacon and the AP buffers the frames till the next beacon, while the client expends a constant current for the duration of T_{BCN}. For a given *timeToNextBeacon*, the average current increases only at the discontinuities described above due to the rounding up effect of PSM. Similarly, for a given *RTT*, we observe that the average current increases linearly with *timeToNextBeacon* and exhibits a jump at the points where the discontinuity condition satisfies. Average current depends upon both *timeToNextBeacon* and *RTT* in case of PSM.

The average current of LTS-PSM is observed to be independent of both *timeToNextBeacon* and *RTT* because LTS-PSM enters the SLEEP state while waiting for TCP ACKs and does not draw extra current while doing so. The current consumption of dPSM and LP-dPSM is found to be independent of *timeToNextBeacon* since they switch to an *Active* profile for TCP traffic. In addition, dPSM and LP-dPSM follow very similar current consumption profiles with average current increasing linearly with RTT. However, dPSM follows a much higher slope due to the higher current drawn while waiting for TCP ACKs. Finally, LP2-dPSM exhibits an average current that is independent of both *timeToNextBeacon* and RTT since an Active profile is used for TCP traffic and the SLEEP state is used while waiting for TCP ACKs.

By uniformly averaging across *timeToNextBeacon*, we arrive at Fig. 10 which represents the realistic case since, in general, TCP transmissions are expected to happen at any *timeToNextBeacon*. We observe that PSM is lower bounded by LP-dPSM, LP2-dPSM, and LTS-PSM. However, we note that LP-dPSM represents a more meaningful limit since it can be construed as the case when PSM enters *SLEEP_BUFFER* only for RTT instead of $\lceil RTT \rceil_{BCN}$. In addition, we observe that LP-dPSM and LP2-dPSM serve as lower bounds for dPSM as the *ACTIVE* current consumption approaches *SLEEP* and *SLEEP_BUFFER*, respectively.

Optimal strategy for known RTT: For a module with the hardware/software modifications that enable all five strategies, the power consumption model can be used to optimally choose power profiles depending upon network RTT conditions. For instance, between PSM and dPSM, it is power efficient to operate with dPSM when network RTT is $< 18ms$. Similarly, it is power efficient to use LP2-dPSM (or LP-dPSM) over LTS-PSM when RTT is $< 10ms$. We observe that LTS-PSM is the overall optimal strategy for $RTT > 15ms$ (with LP2-dPSM having comparable power consumption).

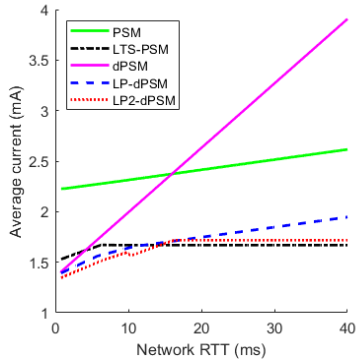


Figure 10: Averaging timeToNextBeacon

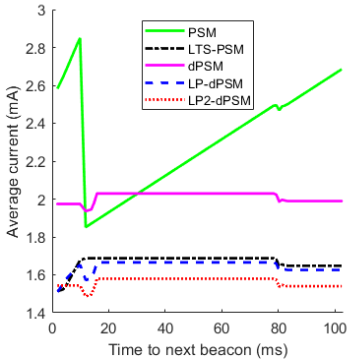


Figure 11: At RTT = 10 *ms*

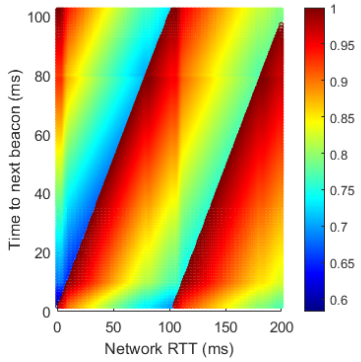


Figure 12: Normalized average current across RTT

PSM and its relation to timeToNextBeacon: Fig. 11 shows the average current consumption for RTT of 10 *ms*. We observe that the average current of PSM varies between 1.85 *mA* and 2.85 *mA* depending on *timeToNextBeacon* while that of other strategies remains relatively constant. The least current consumption occurs at the minimum positive value of

$$timeToNextBeacon = RTT - (K \times T_{BCN}) + \tau, \tag{2}$$

where $K = 0, 1, 2, \ldots$ and τ is a small time interval accounting for delays in frame transmission, AP processing, etc. This is verified in Fig. 12 which shows the plot of normalized average current within each RTT value against *timeToNextBeacon*. Note that the minimum positive value of (2) is the optimal *timeToNextBeacon* value for transmitting a TCP segment.

6 CROSS-LAYER OPTIMIZATION FOR SPARSE IOT TCP UPLINK TRAFFIC WITH PSM

The simulation results from Sec. 5 show that it is most power-efficient to operate the IoT module in LTS-PSM and LP2-dPSM optimization strategies because of the independence of power consumption of RTT and TCP transmission timing. Compared with PSM, LP-dPSM is favorable because of its lower average current and independence of TCP transmission timing. However, to realize the attractive power efficiency performance of LTS-PSM, LP-dPSM, and LP2-dPSM, hardware/firmware changes are necessary. In addition, the power saving using these strategies depends upon the particular device of interest. In this section, we develop a cross-layer power saving optimization that utilizes PSM for single-user sparse uplink IoT TCP traffic by optimally timing TCP segment transmissions with reference to beacons and evaluate it through simulations.

6.1 Algorithm Description

We consider an IoT client with sparse periodic uplink TCP traffic transmitting one TCP segment at a time and focus on the case where the time period between TCP segments T_{data} is sufficiently long such that there is at most one unacknowledged TCP segment at any time instant to demonstrate and evaluate the algorithm ($T_{data} > RTT_{eff}$). When we assume constant RTT network conditions, the optimal time to transmit a TCP segment can be estimated by the minimum positive value of (2). For this scheme, we utilize the beacon timer ($timer_{BCN}$) which counts down from T_{BCN} to zero between beacon receptions.

Algorithm 1: Optimal sparse TCP uplink transmission

Input: μ_{RTT}, σ_{RTT}, T_{BCN}, $timer_{BCN}$, Υ, χ, τ

$RTT_\Upsilon = \mu_{RTT} + \sigma_{RTT} \times \sqrt{2}\,\mathrm{erf}^{-1}(2\Upsilon - 1)$
while *true* **do**
  while *No TCP segment to transmit* **do**
    wait()
  K = 0
  while $[RTT_\Upsilon - (K \times T_{BCN}) + \tau] > 0$ **do**
    $K = K + 1$
  $t_{transmit} = RTT_\Upsilon - (K-1) \times T_{BCN} + \tau$
  if $t_{transmit} < timer_{BCN}$ **then**
    $wait(timer_{BCN} - t_{transmit})$
  else
    $wait(timer_{BCN} + T_{BCN} - t_{transmit})$
  transmit(TCP segment)
  $wait(t_{transmit} + (K-1) \times T_{BCN} + \chi)$
  if *TCP Segment Unacknowledged* **then**
    transmit(PS-POLL frame)

Practical considerations: While the assumption of constant network RTT may hold true to some extent within some local area networks, it is generally not true in the case of the Internet [1]. To account for the variability in RTT, we assume a normal distribution:

$$RTT \sim \mathcal{N}(\mu_{RTT}, \sigma^2_{RTT}), \tag{3}$$

where μ_{RTT} and σ_{RTT} are the mean and standard deviation of the RTT estimated by the IoT module. The RTT model parameters may be updated periodically or through a threshold mechanism. We introduce a design parameter $\Upsilon \in [0.5, 1)$ to determine the Υ^{th} percentile RTT (RTT_Υ) as:

$$RTT_\Upsilon = \mu_{RTT} + \sigma_{RTT} \times \sqrt{2}\,\mathrm{erf}^{-1}(2\Upsilon - 1), \tag{4}$$

where erf^{-1} is the inverse error function (which can be approximated with techniques such as [5]). RTT_Υ will be used to determine the time to transmit TCP segments with a probability $1 - \Upsilon$ of transmitting at the time instants later than the optimal time instant that result in the IoT module waiting an additional T_{BCN} for the TCP

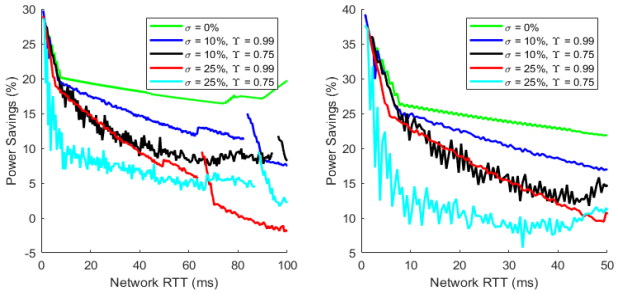
ACK. Υ is lower bounded by 0.5 to avoid RTT_{Υ} being negative or less than μ_{RTT} since $RTT_{\Upsilon=0.5} = \mu_{RTT}$.

We also consider unsuccessful beacon receptions which can possibly lead to higher RTT_{eff}. This issue can be mitigated by sending a PS-POLL frame regardless of beacon reception, anticipating a buffered frame at the AP (as proposed in LT-PSM). With these two enhancements, Algorithm 1 enables the power saving mechanism to transmit TCP segments at the optimal time instant and to transmit the PS-POLL frame after $RTT_{eff} + \chi$ where χ is a short time interval to allow the arrival of TCP ACK in case of successful beacon reception. It should be noted that the case of $\sigma_{RTT} = 0$ corresponds to a constant network RTT.

6.2 Evaluation

We evaluate the proposed optimization algorithm for different μ_{RTT} values with $\tau = 1ms$. The metric used for evaluation is the *power saving percentage* which is defined as the percentage reduction in average current consumption when using the algorithm with a set of parameters in comparison with the average case (TCP transmission at random times). σ_{RTT} is specified as a percentage of μ_{RTT}. Figs. 13a and 13b show the power saving for traffic profiles with 1 (period = 1 s) and 5 (period = 200 ms) TCP segments per second, respectively.

We observe the general trend that with higher RTT, the power saving percentage decreases, demonstrating the higher effectiveness of the algorithm for smaller RTT values. Accordingly, when T_{data} is 200 ms, the power saving percentage is 39%, 31%, 26%, and 24% as μ_{RTT} is <1 ms, 5 ms, 10 ms, and 25 ms, respectively, with $\sigma_{RTT} = 0$, which imposes the higher bound on possible power saving using this algorithm. As the variability of RTT increases, we observe a drop in power saving (comparing $\sigma_{RTT} = 10\%$ and $\sigma_{RTT} = 25\%$ at $\Upsilon = \{0.99, 0.75\}$). Similarly, as Υ decreases for a given σ_{RTT}, a drop in power saving is observed. This may be attributed to the overestimation of RTT_{Υ} at higher σ_{RTT} values and lower Υ values, resulting in earlier-than-optimal transmission of TCP segments.



(a) 1 Segment per second (b) 5 Segments per second

Figure 13: Power saving percentage vs. mean RTT

7 CONCLUSION AND FUTURE WORK

This paper focuses on the low-power operation of Wi-Fi enabled IoT client devices for the sparse uplink traffic scenario. To that end, we present extensive experimental results with a state-of-the-art IoT module to draw key insights. By developing a detailed power consumption model, we present and analyze five strategies through simulations and finally present a cross-layer algorithm to optimize power consumption at IoT client devices compliant with IEEE 802.11 standards that can work with any WFA certified AP.

We plan to extend this work by calibrating the power consumption model to other IoT modules and analyzing the effectiveness of the proposed power saving optimization algorithm on multi-user scenarios. The sensitivity of the proposed optimization algorithm to mismatch between the modeled RTT distribution parameters and network RTT conditions is also a subject for future study.

Acknowledgements: This work was funded in part through a gift from Texas Instruments, the Wayne J. Holman chair, and the National Science Foundation under grant CNS-1813242.

REFERENCES

[1] Jay Aikat, Jasleen Kaur, F Donelson Smith, and Kevin Jeffay. 2003. Variability in TCP round-trip times. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. 279–284.

[2] Unai Alvarado, Guillermo Bistué, and Iñigo Adin. 2011. *Low power RF circuit design in standard CMOS technology*. Vol. 104. Springer Science & Business Media.

[3] Giuseppe Anastasi, Marco Conti, Enrico Gregori, and Andrea Passarella. 2008. 802.11 power-saving mode for mobile computing in Wi-Fi hotspots: limitations, enhancements and open issues. *Wireless Networks* 14, 6 (2008), 745–768.

[4] Dmitry Bankov, Evgeny Khorov, Andrey Lyakhov, and Ekaterina Stepanova. 2019. IEEE 802.11ba — Extremely Low Power Wi-Fi for Massive Internet of Things — Challenges, Open Issues, Performance Evaluation. In *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. 1–5. https://doi.org/10.1109/BlackSeaCom.2019.8812785

[5] JM Blair, CA Edwards, and JH Johnson. 1976. Rational Chebyshev approximations for the inverse of the error function. *Math. Comp.* 30, 136 (1976), 827–830.

[6] Gustavo Carneiro. 2010. NS-3: Network simulator 3. In *UTM Lab Meeting April*, Vol. 20. 4–5.

[7] Feng Chen, Isabel Dietrich, Reinhard German, and Falko Dressler. 2009. An energy model for simulation studies of wireless sensor networks using OMNeT++. (2009).

[8] M Sheik Dawood. 2013. A survey on energy efficient modulation and coding techniques for wireless sensor networks. *Journal of Global Research in Computer Science* 4, 1 (2013), 63–66.

[9] Der-Jiunn Deng, Ying-Pei Lin, Xun Yang, Jun Zhu, Yun-Bo Li, Jun Luo, and Kwang-Cheng Chen. 2017. IEEE 802.11ax: Highly Efficient WLANs for Intelligent Information Infrastructure. *IEEE Communications Magazine* 55, 12 (2017), 52–59. https://doi.org/10.1109/MCOM.2017.1700285

[10] Jie Ding, Mahyar Nemati, Chathurika Ranaweera, and Jinho Choi. 2020. IoT Connectivity Technologies and Applications: A Survey. *IEEE Access* 8 (2020), 67646–67673.

[11] Simon Kellner, Mario Pink, Detlev Meier, and Erik-Oliver BlaB. 2008. Towards a Realistic Energy Model for Wireless Sensor Networks. In *2008 Fifth Annual Conference on Wireless on Demand Network Systems and Services*. 97–100. https://doi.org/10.1109/WONS.2008.4459362

[12] Ronny Krashinsky and Hari Balakrishnan. 2002. Minimizing energy for wireless web access with bounded slowdown. In *Proceedings of the 8th annual international conference on Mobile computing and networking*. 119–130.

[13] Florian Metzger, Tobias Hoßfeld, André Bauer, Samuel Kounev, and Poul E Heegaard. 2019. Modeling of aggregated IoT traffic and its application to an IoT cloud. *Proc. IEEE* 107, 4 (2019), 679–694.

[14] Ugur Olgun, Chi-Chih Chen, and John L. Volakis. 2012. Efficient ambient WiFi energy harvesting technology and its applications. In *Proceedings of the 2012 IEEE International Symposium on Antennas and Propagation*. 1–2. https://doi.org/10.1109/APS.2012.6349384

[15] Brian Peck and Daji Qiao. 2015. A Practical PSM Scheme for Varying Server Delay. *IEEE Transactions on Vehicular Technology* 64, 1 (2015), 303–314. https://doi.org/10.1109/TVT.2014.2319241

[16] George F Riley and Thomas R Henderson. 2010. The ns-3 network simulator. In *Modeling and tools for network simulation*. Springer, 15–34.

[17] Markus Tauber and Saleem N Bhatti. 2012. The effect of the 802.11 power save mechanism (PSM) on energy efficiency and performance during system activity. In *2012 IEEE International Conference on Green Computing and Communications*. IEEE, 573–580.

[18] He Wu, Sidharth Nabar, and Radha Poovendran. 2011. An energy framework for the network simulator 3 (ns-3). In *Proceedings of the 4th international ICST conference on simulation tools and techniques*. 222–230.

[19] Liqiang Zhao, Jian Cai, and Hailin Zhang. 2011. Radio-Efficient Adaptive Modulation and Coding: Green Communication Perspective. In *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*. 1–5. https://doi.org/10.1109/VETECS.2011.5956139