Low-power Internet connectivity over Wi-Fi®

Texas Instruments

Eli Dekel
Wireless system architect
Texas Instruments

In the rapidly growing Internet of Things (IoT), many applications, from personal electronics to industrial machines and sensors, get wirelessly connected to the Internet. These applications cover dozens of use cases, in various environments and serve diverse requirements; no single wireless standard can adequately prevail.

Wi-Fi® is the most ubiquitous wireless Internet connectivity technology today. Its energy consumption and complexity used to be a major barrier for Internet of Things developers, but new silicon devices and modules reduce a lot of the barriers for achieving a low-power Wi-Fi solution and enabling Wi-Fi integration into emerging IoT applications and battery-operated devices. The energy consumption challenge is to ensure that the connectivity technology add-on doesn't impose a change to the power supply. The challenge directly related to energy management is attributed to the fact that communication introduces inherent overheads.

In addition, the average Wi-Fi device energy consumption is significantly impacted by both the access point (AP) behavior and the network/cloud/server performance. Many of the system parameters and performance are independent and given for a certain network. Thus, optimizing Wi-Fi energy consumption while maintaining the system robustness demands a complete understanding of the platform's application, the traffic model and the network properties; these are tied to the application, network, transport and link layers of a networking communication model.

For battery-powered products, one of the main challenges is using the appropriate battery size to achieve an acceptable user experience for battery replacement or recharge period. Certain measures can be taken to favor battery life over other performance aspects. In those cases, it is important to provide the system some guidance as to the target outcome. Extending battery operated product life time requires optimization of the average energy consumption and not just the peak power consumption.

This paper explores the contributing factors to energy consumption, as well as the underlying design considerations. Ways to realize the dominant factors and achieve overall lower energy consumption for typical IoT applications shall be demonstrated. In addition, indirect ways to impact the Wi-Fi lower layer behavior and reduce the total energy consumption of the solution will be described.

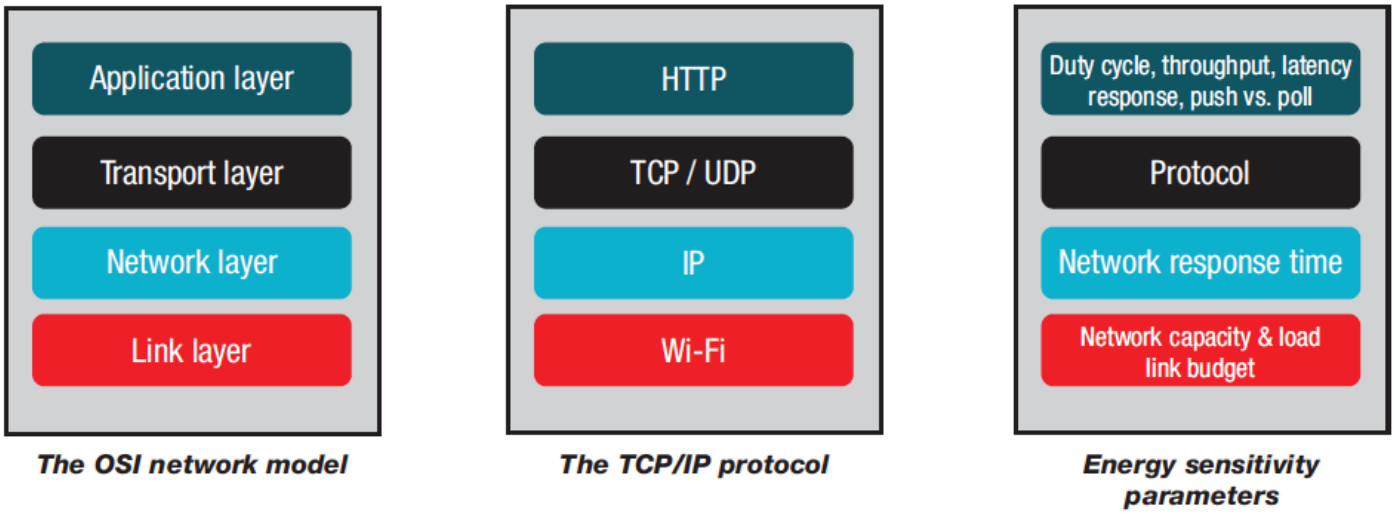

Figure 1. *Simplified OSI model (left) and an example of a TCP/IP protocol stack (right)*

Communication protocols

Communication systems utilize a set of rules and standards to format data and control data exchange. The most common model in data communication systems is the Open Systems Interconnection model (OSI), which breaks the communication into functional layers allowing for easier implementation of scalable and interoperable networks. The OSI model has several layers; a four-layer simplified version of the model is shown in **Figure 1** along with an example of the Transmission Control Protocol (TCP)/Internet Protocol (IP) protocol stack over Wi-Fi.

The **link layer** provides conversion of bits to radio signals (and vice versa), takes care of data framing for reliable wireless communication and manages the access to the radio channel. In the TCP/IP example in **Figure 1**, Wi-Fi is shown as the link layer protocol.

The **network layer** takes care of addressing and routing data through network. The IP is the network layer protocol of the Internet, providing an IP address to devices and carrying IP packets from one device to another.

The **transport layer** generates communication sessions between applications running on two ends of the network. It allows for multiple applications to run on one device, each using its own communication channel. TCP and User Datagram Protocol (UDP) are the transport protocols used in this layer.

The **application layer** is responsible for data formatting and it governs the data flow in an optimal scheme for specific applications. A popular application layer protocol in the TCP/IP protocol stack is Hypertext Transfer Protocol (HTTP) which was created to transfer web content over the Internet.

Robust and scalable IoT solutions are usually based on a layered network implementation, although, for some applications, parts of the layers can be optimized to reduce energy consumption. The behavior of each layer impacts the total application energy consumption.

Energy sensitive parameters

The IoT product energy consumption can be analyzed by the traffic model, network properties and platform constraints (see Figure 2 on the following page). The average energy consumption profile varies due to these parameters. The following section describes the parameters which impact the system energy consumption for each discipline.

The application energy consumption profile describes the consumption of the system for

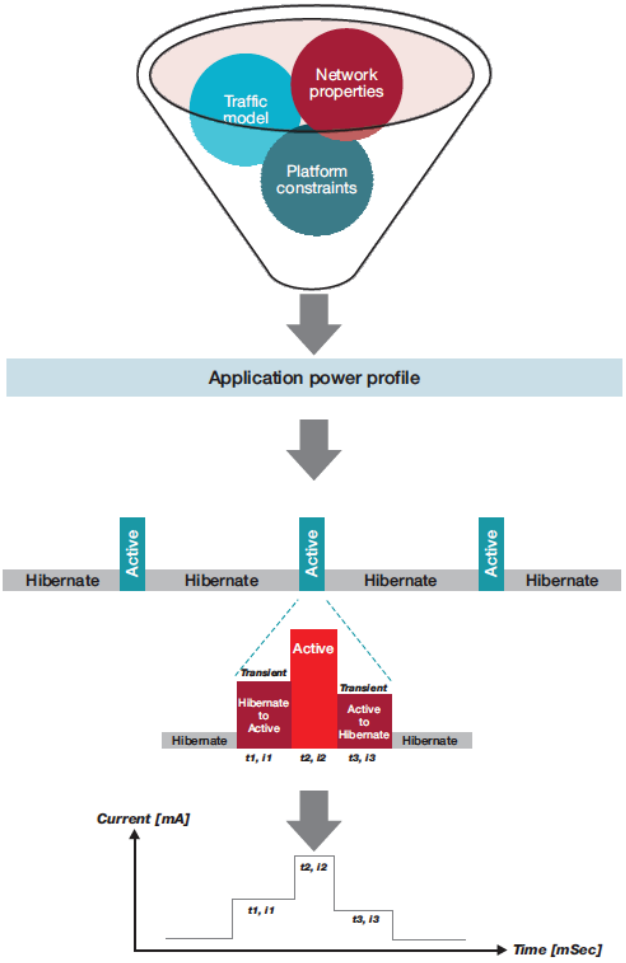

Figure 2. *Application energy consumption profile analysis flow*

different system modes over time. The profile also includes the energy consumed for the transition between system modes. The total energy consumed is an integral of the current consumed, measured in units of amperes, over time from the supply:

$$\text{Energy: } E\,[Jouls] = \int Vsupply * I\,dt$$

Alternatively the charge consumption could be calculated:

$$\text{Charge: } C\,[coulomb] = \int I\,dt$$

Commonly, a simpler, piecewise linear approximation description of energy consumption is used:

$$\text{Energy: } E\,[Jouls] = \sum_{x=1}^{n} Vsupply * I_x * T_x$$

$$\text{Charge: } C\,[coulomb] = \sum_{x=1}^{n} I_x * T_x$$

The capacity of battery tells how much charge it can store. Capacity can be specified by ampere-hour rating which is equivalent to charge. The application charge consumption and the battery capacity determines the application life time.

Traffic model

The traffic model describes the data or packet flow over the network. The packet flow is driven by the average throughput required for the application, the required response latency, activity duty cycle and whether data is transmitted in burst mode or continuously. The traffic model is application specific and its characteristics are mostly dominated by the use case. The following sections describe the traffic model parameters.

Protocol

The protocol refers to the packet/message format and the handshake with the server. Every packet transferred over the network includes overhead bits. The packet format includes the packet header and the actual data that needs to be transferred over the network.

The TCP and UDP are transport layer protocols used for sending data over the Internet. TCP is a connection-oriented protocol which requires establishing a connection prior to transmission. It provides reliable delivery of messages and in the right order. TCP overhead is higher due to the bigger header and the requirements for data acknowledgment, resulting in an increase

TCP segment header format					
Bits	0 7 8	15 16	23 24	31	
0	Source port		Destination port		
32	Sequence number				
64	Acknowledgement number				
96	Data offset	Reserved	Flags	Window size	
128	Header & data checksum		Urgent pointer		
180	Options				
UDP datagram header format					
Bits	0 7 8	15 16	23 24	31	
0	Source port		Destination port		
32	Length		Header and data checksum		

Table 1. *TCP and UDP header formats*

in energy consumption. UDP is a connectionless protocol with no connection setup required prior to transmission. UDP supports best effort delivery of message, there is no guarantee that the messages sent would reach at all. UDP protocol requires smaller overhead and no response or acknowledgment (ACK) from the server. This results in less energy consumption. The protocol used need to take into account the application requirements and the impact on the overall energy consumption. The most efficient protocol should be used.

Throughput

The application throughput refers to the amount of data passing through the network in a given period. The throughput requirement definition includes the amount of data and its distribution over time. The amount of data impacts the active period that the system needs to communicate with the network while the distribution can affect the sleep period and the network behavior. The data size could be constant per activity or it may vary over time. The distribution of the data can be periodic when activity occurs sporadically over time, variable when the activity is not periodic, or one-shot when data is sent once.

Latency response

Latency is the time interval between an event and a response. It defines how fast the system should respond to an event. An event could be an external trigger from a peripheral, access from the network, a change in network status, etc. A response could be activating a peripheral, sending a notification to the network, etc.

The application defines the required latency response. The latency affects the system maximum sleep period, whether the IoT device needs to stay connected to the network or reconnect after a predefined period, etc.

For instance, in a temperature-sensor application that updates the server with the measured temperature every few minutes, the latency response time is practically not limited. The application could work in low-duty cycle and optimize its energy consumption; however, if the temperature-sensor application is required to immediately notify when the temperature passes a certain threshold, the latency response time should be extremely small. In this case, the application should stay connected to the network; this will result in a significant amount of energy consumption. The right tradeoff needs to be taken to make sure the

application meets its requirement while still having low Wi-Fi energy consumption operation.

Push vs. pull mode

An application can obtain data from a server in push or pull mode. In the case of push mode, the server pushes the data to the IoT device and the IoT device needs to be ready receiving the data from the server. In this mode, the IoT device must stay associated with the Wi-Fi network and keep an active channel with the server; however, in case of pull mode, the IoT device can select whether to stay connected to the server or re-connect when data needs to be retrieved. This can reduce the energy consumption significantly.

One should note that at the traffic model level this is applicable to the application layer and should not be confused with the fact that other layers may – implicitly or explicitly – apply a different method for various reasons. Such is the case with a push application that may use Wi-Fi protocol power-save delivery scheme for the sake of reducing power at the link layer. In this case, the link layer will work in pull mode, polling the access point, while the application may still be operating in push mode from an application layer perspective.

Burst transfer

Burst transfer refers to events that occur in rapid succession. Depending on the throughput and latency response requirements, the system can define the data transmission profile. For instance, the system can communicate with the network whenever data is available or accumulate the data and communicate with the network after passing a certain threshold or period of time. Accumulating data enables the system to sleep for a longer period and reduce the number of transitions between active and sleep. In addition, it reduces the protocol overhead for every communication with the server.

Duty cycle

The duty cycle is the percentage of time the system is active. Reducing the duty cycle reduces the system energy consumption. The application throughput and latency requirements define the system duty cycle. For instance, higher throughput and low latency requirements increase the duty cycle.

Platform constraints

The platform constraints refer to the boundaries associated to the system such as the total solution size, power supply type, voltage level and antenna size, etc. The following sections describe the main system limitations which affect the energy consumption and battery life of the product.

Power supply

The power supply delivers the energy used by the system. It can obtain its energy from various types of sources such as batteries, AC power and powerharvesting cells. The power supply type impacts the total energy that can be consumed and the peak power. For instance, batteries are available in different sizes and capacities. Each battery type supports a different capacity value and max peak current. AC power applications have “unlimited” energy with no real limitation on the average energy consumption of the load.

Total area solution

In many cases the product has to meet a certain size and total area solution. This could impact the antenna size and its location, etc. The antenna size, gain and type impact the energy required to transfer a data over the network. Smaller antenna size or

gain requires transmitting at higher output power to compensate for the antenna size.

Network properties

A variety of network parameters may impact the overall system behavior and its energy consumption. These include network capacity, network load, link budget, response time, etc., and are mostly uncontrollable by the application.

Network capacity & load

Network capacity is a measurement of the max amount of data that can be transferred between network locations over a link. An increase in network usage increases the communication period over the network and therefore the overall energy consumption of the device. Note, the number of clients over the network and the required throughput affect the network load and the response time. A reduction of the network usage for a specific throughput can be done by reducing the number of transmissions and their duration. The number of transmissions is reduced by aggregating data up to the max packet allowed (up to the latency requirement). The transmission duration can be reduced by using a higher Wi-Fi rate. The rate is usually determined by the access point and the device in the link layer for achieving optimized radio, energy consumption and throughput performance; thus uncontrollable by the application.

Link budget

The link budget is a measure of a wireless communication system link quality. Low-quality link impacts the number of times a packet needs to be sent (e.g., retransmission) and thus the overall energy consumption. The transmitted packet output power and the rate defines the packet error rate over a channel and the energy consumption. There is a clear tradeoff between improving the link quality and the energy consumption.

Network response time

Response time is the total amount of time it takes to obtain a response for a request sent to another destination in the network, e.g., a server. Many factors affect this time such as the network capacity, network load, link budget, etc. The application energy consumption is significantly impacted by the network response time, and therefore might increase the energy consumption. For instance, an IoT device that communicates with the server using TCP/IP protocol will wait for responses over the TCP/IP layer. These responses depend on the network topology, capacity, traffic and the protocol. This delay can vary from 10s to 100s of msec.

The behavior of the IoT device, while waiting for the server's response, significantly impacts the total energy consumption. On one hand, the IoT device could go to its lowest power mode to save power and wakeup periodically to check if the response has been arrived. This increases the latency response in the application layer. On the other hand, the IoT device can stay active to receive the response immediately when it arrives.

There are many methods for tradeoff energy optimization and system robustness. One viable method to save energy is specific to the link-layer protocol used. Such as the power-save mode in Wi-Fi (802.11) which enables the IoT device to inform the access point (AP) it is in power save. From this point onwards, the AP buffers traffic for the IoT device and allows the device to poll for this traffic at will. The indication for buffered traffic is available in specific beacon packets. The device can be programmed to wake up only for these

packets thereby saving the energy in between. The main disadvantage of this method is that the communication between the server and IoT device is limited to the wakeup interval of IoT device. This practically means increasing the latency. For latency-sensitive applications this method is not applicable.

Profile derivation guidelines/ methodology

To generalize the concept from a case study to a guideline, the following recipe can be followed. A fairly simple, step-by-step procedure is provided to allow realizing pretty complex schemes. The needed input to the scheme is a complete description of the analyzed profile. To realize the methodology, it is necessary to align the various power modes of the system.

System power modes

The energy consumption profile describes the combination of various system modes over time. The following section describes common system modes for a wireless system. It is noted that there is no standard terminology, and one should be well aware of the exact behavior in each mode in order to allow proper understanding of the entailed overheads per mode.

The following terminology is consistent with TI's SimpleLink™ Wi-Fi family of products, and it is provided here for clarity of the discussion hereinafter.

- **Shutdown -** the device is in its lowest energy consumption mode without keeping its state or parameters and with RTC clock turned off. Exit from this mode requires waking up of the RTC and then full initialization of the device. This mode should be used when no Wi-Fi connection is required and that long init time is acceptable (>1sec).
- **Hibernate -** the device is in a very low energy consumption mode without keeping its state or parameters (RTC clock could be on). Exit from this mode requires full initialization of the device. Device can enter this mode when no Wi-Fi connection is required.
- **Sleep -** the device is in low-power mode but keeping system state and parameters in retention. The device can enter this mode between network communications while keeping the Wi-Fi network and TCP IP parameters and state.
- **Idle -** (or standby) – the device is active but without radio activity. For instance, device which wakes up to idle to perform temperature measurement.
- **Active -** the device is communicating with the network and transmitting or receiving. This mode consumes the most power.
- **Transition -** between modes consumes energy as well and needs to be taken into account when calculating the total energy required for the use case.

Methodology

The following example shall be used throughout this recipe to demonstrate each step – "a disposable wrist gadget, powered by a 120 mAh, 3V Li-Poly battery is envisioned. The device is always connected, listening to beacons every 1 sec for downlink information and updates its status every minute. During manufacturing the device is powered in order to run a series of qualification tests, which last 20 msec; at which time the device is fully active. The likelihood of downlink information is rare and may be neglected (once an hour). Between activities the device enter low power sleep mode (LPDS). What is the lifetime of the wrist gadget?"

Achieving a low-power Wi-Fi application requires understanding each one of the application energy consumption contributors. This includes the contribution of the activities which are done once or rarely and the activities which are done periodically. Without further ado, here it is:

All Energy consumption numbers below are based on the average energy consumption of available solutions in the market.

Steps	Description
Step 1. Decompose use case into well-defined, independent traffic profiles	• Beacon reception every 1 sec • Status updates every 60 sec • Power up twice (for production + in field) • Manufacturing qualification tests
Step 2. Split the profiles into one-shot activities, and periodic activities and enumerate each group, denoting them { o_i }, { p_i } respectively	O_1: Power up O_2: Manufacturing qualification tests P_1: Beacon reception P_2: Status updates P_3: Low-power sleep mode
Step 3. Calculate total charge spent on one-shot activities denoted by C_0 (see Step 0.1) C	$C_0 = O_1 + O_2 = 50 + 400 = 450$ µC = 450 mAmS
Step 4. Calculate average current for periodic activities denoted by IP (see Step P.1)	$I_P = I_{P1} + I_{P2} + I_{P3} = 3 + 8.3 + 100 = 111.3$ µA
Step 5. Calculate lifetime as follows: Lifetime = $(C_{BAT} - C_0) / I_P$	C_{BAT} = 120 mAh Lifetime = (120 mAh – 450 mAmS) / 111.3 µA = ~1.6 months
One-Shot Activity – Energy Calculation	
Step 0.1. For each activity in { o_i }, denote its duration and current, t_i and I_i respectively	O_1: I_1 = 5 mA; t_1 = 10 msec O_2: I_2 = 20 mA; t_2 = 20 msec
Step 0.2. Calculate the charge spent per activity $C_i = t_i * I_i$	$O_1 = I_1 * t_1 * 2 = 5$ mA * 10 msec = 50 µC $O_2 = I_2 * t_2 * 1 = 20$ mA * 20 msec = 400 µC
Periodic Activity – Average	
Step P.1. For each activity in { p_i }, denote its duration and current, t_i and I_i respectively	P_1: I_1 = 30 mA; t_1 = 0.1 msec; period =1 sec P_2: I_2 = 50 mA; t_2 = 10 msec; period = 60 sec
Step P.2. Calculate the charge spent per activity $I_P = \Sigma t_1 * I_i$ / period	$I_{P1} = I_1 * t_1$ / period = 30 mA * 0.1 msec / 1 sec = 3 µA $I_{P2} = I_2 * t_2$ / period = 50 mA * 10 msec / 60 sec = 8.3 µA I_{P3} = 100 µA

Case study – temperature sensor

In this section, a practical use case will be analyzed including the impact of each one of the main parameters on the overall system energy consumption.

The analysis is done for a temperature sensor application which measures the temperature every 60 sec and communicates it to a server – "a temperature sensor, powered by a 3000 mAh, 3V Li-Ion battery is envisioned. The device is always connected, listening to beacons every 300 msec for downlink information and updates its status every minute. The likelihood of downlink information is rare and will be neglected (once an hour)."

The **Table 3** describes the main properties of the use case.

Power profile

In this use case the system wakes up every 60 seconds to send the temperature measurement result to a server. Between activities the system need to stay associated with the Wi-Fi access point (AP) and listen to its beacons (we will assume beacon interval of 100 msec with DTIM = 3). The rest of the time the system could be in sleep mode.

In addition, the system needs to associate with the Wi-Fi AP and open TCP/IP socket with the server. This could be done once in the beginning of life and the energy consumed will be ignored in this analysis.

Paramater	Value
Protocol	TCP IP
Throughput	64 Bytes, every 60 sec
Latency response NA	NA
Burst	No burst – single temperature sample transmitted every 60 sec
Duty cycle	Temperature sample: <1/60 Listening to AP every 300 ms: Beacon interval of 100 ms and DTIM = 3
Power supply	3V, 3000 mAh battery
Total area solution	NA
Network capacity & load	No limit
Link budget	No error
Network response time	100 ms

Table 3. *Temperature-sensor application parameters*

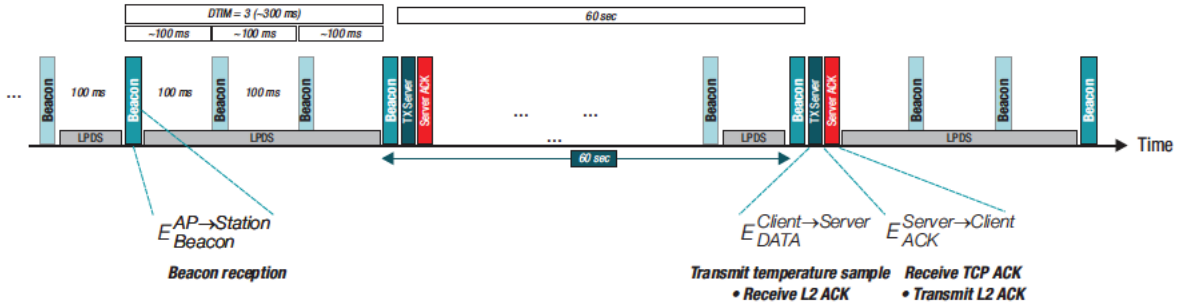

Figure 3. *Temperature sensor power profiles and system modes*

In the energy consumption analysis, both the energy consumed in each mode and the energy consumed for the transient between modes will be included in the overall energy consumption.

Figure 3 describes the energy consumption profile for this scenario after the application is associated with the AP and has an open TCP/IP socket with the server:

Since the use case is periodic, the average energy consumption is calculated for a 60-sec period.

Every 60 sec the system consumes energy for:

- $E_{\substack{AP\rightarrow Station\\Beacon}}$ – Energy consumed for receiving a single beacon

  including the reception of the beacon and the transient from LPDS and RX modes and vice versa. Beacon is sent every 100 msec however in this use case DTIM=3 is used which the station listen to beacon every three beacon periods e.g., every 300 msec. The total energy includes the transitions from LPDS to RX, the reception and the transition back from RX to LPDS.

- $E_{\substack{Client\rightarrow Server\\DATA}}$ – Energy consumed for sending packet to the server including the transmission of the temperature sample and receiving L2 ACK from the AP.

  The energy includes the transients from LPDS to transmit mode and from transmit to receive.

  Sending packet to server is done once every 60 sec.

- $E_{\substack{Server\rightarrow Client\\ACK}}$ – Energy consumed for receiving TCP IP ACK from the server including reception of the ACK, transition from RX to TX and sending L2 ACK.

Energy parameters	
Use case modes	Description
$E_{\substack{AP\rightarrow Station\\Beacon}}$	Beacon reception energy
$E_{\substack{Client\rightarrow Server\\DATA}}$	Temperature sample sent to server energy (including L2 ACK)
$E_{\substack{Server\rightarrow Client\\ACK}}$	ACK received from server energy (including L2 ACK)
Static modes	Description
E_{LPDS}	Sleep energy

Table 4. *Temperature sensor power profiles parameters description*

Steps	Description
Step 1. Decompose use case into well-defined, independent traffic profiles	• Beacon reception every 300 msec • Status updates every 60 sec • Power up once • Downlink is rare – will be neglected • Low-power sleep mode
Step 2. Split the profiles into one-shot activities, and periodic activities and enumerate each group, denoting them { o_i }, { p_i } respectively	O_1: Power up once – will be neglected O_2: Downlink is rare – will be neglected P_1: Beacon reception $E_{\substack{AP\rightarrow Station\\Beacon}}$ P_2: Status updates (TCP–IP Data) $E_{\substack{Client\rightarrow Server\\DATA}}$ P_3: Status updates (TCP–IP ACK) $E_{\substack{Server\rightarrow Client\\ACT}}$ P_4: Low-power sleep mode
Step 3. Calculate total charge spent on one-shot activities denoted by C_0	C_0 = Negligible
Step 4. Calculate average current for periodic activities denoted by I_p	$I_P{}^1 = I_{p1} + I_{p2} + I_{p3} + I_{p4} = 235 + 4 + 70 + 100 = 409$ µA
Step 5. Calculate lifetime as follows: Lifetime = $(C_{BAT} - C_0) / I_P$	C_{BAT} = 3000 mAh Lifetime = 3000 mAh / 409 µA = ~10 months

Table 5. *Temperature sensor application – Lifetime calculation*

This is done once every 60 sec. The TCP/IP ACK reception period depends on the network response time which can vary significantly and impact the overall energy consumption since the system needs to stay in reception mode during this period.

- E_{LPDS} – Energy consumed during sleep. When not communicating with server and not listening to beacons, the device is in sleep mode.

Table 5 includes a calculation of each one of the parameters. All energy consumption numbers below are based on the average energy consumption of available solutions in the market.

In the example in **Table 5** the main contributor to $(P1: E_{Beacon}^{AP \rightarrow Station})$ the energy consumption is the beacon reception

and the second contributor is the energy consumed $(P3: E_{ACK}^{Server \rightarrow Client})$ during sleep (P4). The energy consumed for receiving TCP ACK

is also significant and consume 17% of the total energy, this is mainly due to the long network response time. Transmitting the

temperature sample to the server $(P2: E_{DATA}^{Client \rightarrow Server})$ does not take a significant part of the total energy consumption. See **Figure 4.**

Energy dependent parameters

This section describes how variance of each one of the parameters impacts the overall energy consumption of the use case.

Protocol selection – UDP vs TCP: If the protocol used is UDP, then TCP ACK is not required and $E_{ACK}^{Server \rightarrow Client}$ could be eliminated and save 17% from the overall energy consumption. The disadvantage is the fact that the system does not get ACK from the server and it might be that data will be lost. For some applications missing data is acceptable.

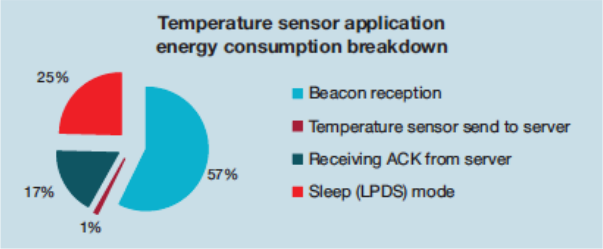

Figure 4. *Temperature sensor application energy consumption breakdown*

Corollary 1. For reliable data prefer TCP. For volatile traffic with minimum overhead prefer UDP.

Throughput 512 B vs 64 B: In case more data needs to be transmitted it will increase the energy for transmission $E_{DATA}^{Client \rightarrow Server}$. However energy consumed for transmitting the temperature sampleis negligible versus the overall energy and remains such up to the granularity of the TCP packet size (1.5 KB). Sending multiple TCP packets is followed by an ACK substantially increasing the energy consumption.

Corollary 2. In lean data scenarios pack transmitted data up to the max TCP packet size.

Burst: Working in burst mode: In case it is acceptable that the application will, for instance, store five samples and send all five every 300 sec (5 × 60 sec). This will reduce the $E_{ACK}^{Server \rightarrow Client}$ since it will done once every 300 sec only reducing the total energy 13.6% (the energy consumed will be 17% / 5). The disadvantage is the fact that temperature sample will be delayed by 300 sec and the last samples in the server are five minutes old in the worst case scenario.

Corollary 3. Aggregate is beneficial up to the latency level.

Network response time: Network response time impacts the $E_{ACK}^{Server \rightarrow Client}$. Longer response time means that the system needs to stay in receive mode longer (instead of sleep) and actually increasing the overall energy consumption significantly. Optimizing the network response time could save significant energy.

Corollary 4. Topology matters. In poor network response time topology energy consumption will rapidly deteriorate.

Beacon skipping: The beacon reception consumed the most significant amount of energy for this use case. It consumes 57% of the total energy. Skipping beacons will reduce the number of wakeups of the system. For instance receiving one of five beacons periodically (DTIM=5, 500-msec period) will reduce the total energy of the use case by ~25 percent. Legacy infrastructure requires a minimal presence to maintain the connection to the AP. However, a new set of Wi-FI alliance specifications called "IoT Low Power" will allow stations to be abscent from the network for a long time (many seconds) while AP connection is guaranteed.

Corollary 5. Maximize the beacon skipping interval to fit application latency.

Connection paradigm: Stay connected versus re-connect – To enable 100% availability, the sensor must stay connected to the AP. This energy will be denoted $E_{Stay\ Connected}$. Alternatively, the sensor can disconnect and reconnect from the network periodically. This energy will be denoted $E_{Reconnect}$.
In the stay connected case, receiving beacons and staying associated with the AP consumes 59% of the total energy budget, yet, it allows 100% network availability and enables the system to enter LPDS mode between activities. It is important to pay attention to the fact that the energy involved in connection and beacon reception is strongly influenced by the AP behavior and network topology. The response times of both the AP and the remote server result in longer duration to complete the activity. As a result the energy expenditure increases.

In a reconnect case, the sensor will not have to wake up for receiving beacons, thus, allowing it to enter hibernate which is ~50× lower power state than LPDS which means a reduction of ~25% from the total energy for this use case. On the other hand, the sensor will have to go through authentication and association processes every wakeup and re-open the TCP/IP session. In addition, this paradigm is adequate for cases not requiring 100% availability for remote paging.
In case the energy to reconnect $E_{Reconnect}$ (where the initialization and connection with AP/server could be the dominant factors) from power up is smaller than the energy to stay connected $E_{Stay\ Connected}$ (where the beacon reception is the dominant factor) then it is better to reconnect instead of staying connected to the AP. **Figure 5** is an example of crossover analysis between the two paradigms.

Corollary 6. In case of a latency-sensitive application or 100% availability is required "always connected" paradigm should be selected; otherwise, in low-duty cycle applications reconnection is a more power efficient paradigm.

Corollary 7. If energy consumption is the sole factor, identify the crossover period in which $E_{Reconnect} = E_{Stay\ Connected}$.

Texas Instruments SimpleLink Wi-Fi devices

Texas Instruments SimpleLink Wi-Fi Interneton-Chip™ devices will be used to calculate and present the tradeoff between "stay connected" vs. "reconnect" paradigm for the sensor use case described above.

SimpleLink™ Wi-Fi CC32xx device family from Texas Instruments™ is a system-on-chip (SoC) solution that integrates two processors within a single chip, including:

- Application processor: Arm® Cortex®-M4 MCU with a user-dedicated 256KB of RAM and an optional 1MB of executable flash
- Network processor to run all Wi-Fi and internet logical layers. This ROM-based subsystem includes an 802.11 a/b/g/n dual band 2.4-GHz and 5-GHz radio, baseband, and MAC with a powerful hardware cryptography engine

The advanced device power management subsystem and policy management enables the user to work either in a "stay connected" method or "reconnect" method. For a "stay connected" method the performance is shown for beacon reception with long sleep interval (LSI) feature which enables skipping beacons for different time intervals, for instance, 0.3 sec (DTIM=3) and with "IoT low power" feature which enables WiFi station to go to sleep for long time duration, for instance 4 sec:
The crossover point for beacon reception every 300 msec is at ~19 sec period. If the application is updating the server with the measured temperature every 5 sec it is better to stay connected; however, if updating the server occurs every 60 sec, it is better to reconnect every wakeup. More information can be found in this application note: [SimpleLink Wi-Fi CC3100/CC3200 Internet-on-a-chip networking subsystem power management.]

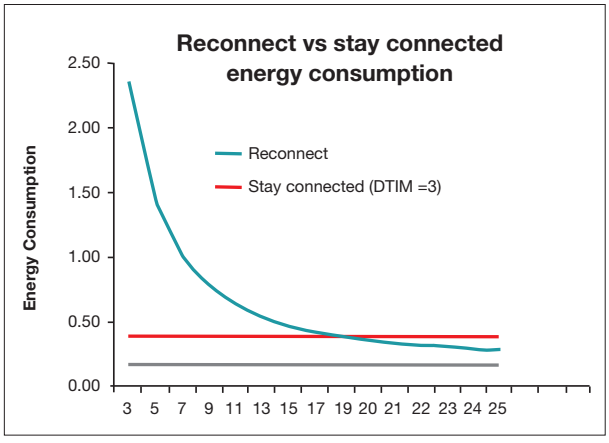

Figure 5. *Reconnect vs. stay connected application energy consumption*

Summary

Low-power Internet connectivity using Wi-Fi is a reality. The challenge is to ensure that the wireless connectivity technology add-on doesn't impose a change to the power supply. For battery-powered products, certain measures can be taken to favor battery life over other performance aspects. In this paper we explored the contributing factors to energy consumption and related design tradeoffs.

A clear methodology for power profile analysis was defined allowing developers to determine how to achieve overall lower energy consumption for typical IoT applications. In addition, ways to impact the Wi-Fi lower layer behavior and reduce the total energy consumption of the solution have been described.

In many typical applications, the amount of information to be transmitted is significantly lower than the total bits over the air. The fact that communication protocols introduce inherent overheads contributes to the total energy consumption. The impact of these overheads on the overall energy consumption could be reduced by managing the traffic properties and profile of the application. As described in the paper, this is a challenging task. Highly integrated networking solutions, like the Texas Instruments SimpleLink Wi-Fi CC3235 wireless MCU, encapsulate capabilities for easily managing traffic properties on-chip and alleviate the complexity of optimizing the application's energy consumption.

The contributing factors to energy consumption can be managed to meet low-power Wi-Fi application goals. The dominant factors for achieving overall lower energy consumption for typical IoT applications needs to be carefully analyzed. The key to a good solution lies in understanding those contributors and properly following the corollaries described in this paper. Deep understanding of the application-level requirements and the target performance enables defining the application behavior over the network and optimizing the average energy consumption while keeping the system robustness.

P1: Beacon reception E			
Period (sec)	**Reconnect**	**Stay Connected (DTIM=3)**	**Stay Connected (IoT lowi power = 4sec)**
1	7.03	7.03	0.156
3	2.34		
5	1.41		
7	1.00		
9	0.78		
11	0.64		
13	0.54		
15	0.47		
17	0.41		
19	0.37		
20	0.35		
21	0.33		
22	0.32		
23	0.31		
24	0.29		
25	0.28		

Table 6. *Texas Instruments SimpleLink Wi-Fi Internet-on-Chip device reconnect vs. stay connected application current consumption*

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265