A Framework for Power Saving in IoT Networks

Mukesh Taneja
Cisco Systems
Bangalore, India
mukesht@ieee.org

Abstract— **An IoT / M2M system may support large number of battery operated devices in addition to some mains operated devices. It is important to conserve energy of these battery operated constrained devices. An IoT / M2M Gateway used in this system is an intermediate node between IoT / M2M devices and an IoT / M2M Service Platform. It enables distributed analytics and helps to reduce traffic load in the network. This gateway could be stationary or mobile. In an IoT / M2M system, it becomes important to conserve energy of this Gateway as well. This paper proposes a framework to reduce power consumption of M2M / IoT devices as well as Gateway nodes. We buffer data at IoT Application, IoT Gateways and Devices to keep devices and Gateway nodes in sleep mode as long as possible. We allow computation of the duration to buffer this data using factors such as QoS requirements, predicted pattern of future IoT / M2M messages and congestion indicators from different network nodes. This potentially also allows intelligent aggregation of IoT messages at the Gateway node. We also enhance signaling mechanisms and present software building blocks for this framework. Mesh as well as Cellular access technologies are considered here.**

Keywords— Internet of Things (IoT); Machine to Machine (M2M) Communication; Power Saving, IEEE802.15.4.

I. INTRODUCTION

The Internet of Things (IoT) is the network of physical objects that can be monitored and controlled through the Internet. Data collected through this network is analyzed to take business decisions. A schematic diagram of an M2M / IoT system is shown in Figure 1. An end device used in an M2M system is represented by M2M / IoT device in this figure. An M2M / IoT Gateway (GW) is an intermediate node that collects and aggregates data from M2M devices. It can run local analytics applications (or participate in running distributed analytics applications). For example, it can analyze periodic temperature readings obtained from a temperature sensor and compute maximum or minimum or average temperature as needed by a user over a time period. It can reduce the amount of traffic in the network by only sending relevant data to a user. An M2M / IoT GW can also help in taking real-time decisions by doing computations closer to edge of the network. If messages are encrypted using Datagram Transport Layer Security (DTLS) or Transport Layer Security (TLS), it becomes difficult to analyze data at IoT GW as it cannot decrypt these messages. To decrypt data at IoT GW, it needs to have access to security keys and it needs to build trust relationship with the IoT Service Platform. [6] Proposes a lightweight framework where security and application protocol proxies are run at the IoT GW to enable this. For example, proxies for Constrained Application Protocol (CoAP) and Datagram Transport Layer Security (DTLS) are run at this Gateway using a lightweight mechanism specified in that paper. An IoT / M2M user gets access to this system via a Service Platform in this architecture. A utility company, a cab company or a car manufacturer (that is providing remote diagnostics service) are some examples of the IoT users that may want to access such a system. This Service Platform could be used for various purposes such as provisioning, firmware upgrade, diagnostics and end-to-end connection (or session) management for devices as well as IoT GW nodes. It could potentially consist of connectivity as well as an application development platform. In the remainder of this paper, we use the word "device" to refer to a device (such as a sensor or an actuator) that is used in an M2M system or in the larger context of Internet of Things. We also use "GW" to refer to a Gateway that is used in an M2M / IoT system.

A device could be a low cost device that is used to measure an environment variable such as temperature or it could be even a high cost device such as a video surveillance camera. Some of these devices could be battery operated while some could be mains operated. In some deployment scenarios, it is not feasible to keep charging (or replacing) battery frequently and it is necessary to have mechanisms in place so that battery of such a device can last for long time. These devices consume high amount of power when they are in active state and low amount of power when in sleep (or idle) state. For some of the 802.15.4 based devices, current drain in active state could be few 10s of mA while it could be few microA in sleep state. Keeping these devices in sleep state help prolong the battery life but it needs to be done in a way so that latency constraints are not violated.

Authors in [2], [3], [4], [7] and [8] propose or analyze some techniques to reduce power consumption of devices. Cellular networks such as LTE networks support Discontinuous Reception / Transmission (DRX/DTX) to allow devices to turn off their radios. Optimization of DRX/DTX parameters to conserve energy is considered in [2]. Authors in [3] propose a paging technique using IoT group id. They also propose removal of tracking and routing area update messages for low mobility devices, and optimize paging related signaling. Impact of extending the paging cycle and reducing the Radio Resource Control (RRC) connected-to-idle transition tail time on power saving in evaluated in [8]. Authors in [4] propose a scheme for IEEE 802.11 networks that grants higher channel access priority for low energy devices dynamically. Authors in

[7] propose a method that spreads the M2M traffic evenly with calculated offsets to alleviate network contention and to reduce packet delay.

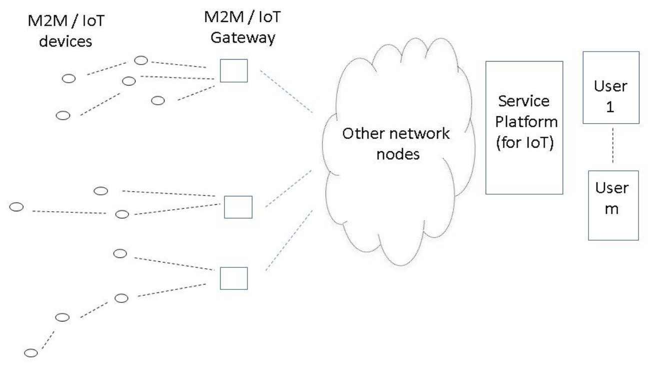


Figure 1: An M2M / IoT System – Reference Architecture

An IoT GW could be stationary or mobile. For example, if a GW is located in a moving object (such as a vehicle or a robot), it also moves along with that object. An example scenario where this GW communicates with various other devices is shown in Figure 2. In this scenario, IoT GW uses cellular backhaul to communicate with network infrastructure nodes. It uses short range technologies such as IEEE 802.15.4 [5] to communicate with other IEEE 802.15.4 based devices. It supports LTE-Direct and WiFi-Direct for device-to-device communication where one of the devices is part of the GW itself. For medium to long range communication, it uses cellular technology. This GW supports User Equipment (UE) as well as Access Point (AP) functionality for Cellular (such as LTE, 3G) and WLAN access technologies as shown in Figure 3. It is desirable to reduce power consumption of this GW also.

Our goal is to reduce energy consumption of devices as well as GW nodes. As devices and GW nodes could support multiple access technologies, we need to have a solution that works with these heterogeneous access technologies. We propose a power saving framework for devices and gateway nodes in the following sections.

II. POWER SAVING FOR M2M / IOT DEVICES

An IoT application running with a user (or IoT service platform) may contact a device for several reasons. It could communicate with the device to configure some parameters or update firmware of the device. It could send a command to device to get some data. For example, it could ask for current location or velocity of the device. If it is an urgent request, it may ask the device to send data immediately. There could be many other requests that are delay tolerant and not urgent. For example, it may be acceptable to a utility company to get reading from an energy meter in 1 min or 10 min or even in some hours after it has issued request to get data. An example delay tolerant request is shown in Figure 4. Here, an IoT request j is generated at time T_j and response needs to be received by time T_{j+1} . For delay tolerant request j in Figure 4, we have

$$t_4 - t_1 \le T_{j+1} - T_j, \quad t_1 \ge T_j$$

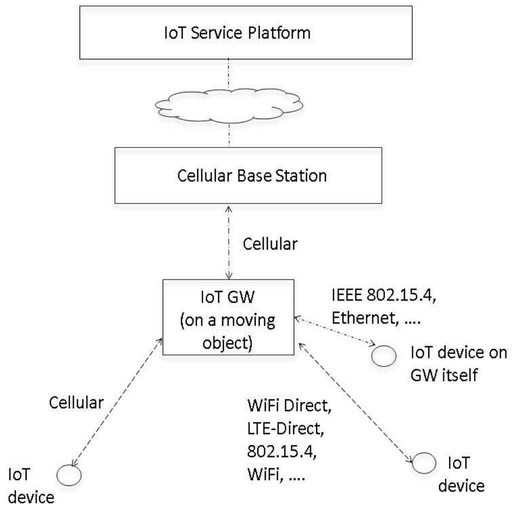

IoT Service Platform
Cellular Base Station
Cellular
IoT GW (on a moving object)
IEEE 802.15.4, Ethernet, ...
IoT device on GW itself
Cellular
WiFi Direct, LTE-Direct, 802.15.4, WiFi, ...
IoT device
IoT device

Figure 2: A Mobile IoT GW (an example scenario)

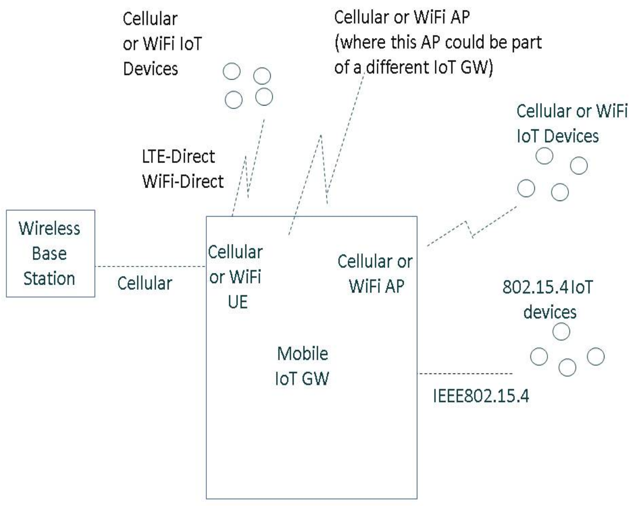


Figure 3: An IoT GW with Heterogeneous Interfaces

To conserve energy for battery operated devices, we want to keep them in the sleep state as long as possible and make them active only when it really necessary. We classify devices in three categories based on the nature of their sleep schedule. First category of devices have relatively stable sleep schedule. Some of these devices may be in sleep mode for most of the time [1]. In second category, we put devices where the sleep schedule may change quite often. For example, a device that provides location or velocity of a moving vehicle may change its state very often. In the third category, we put devices that are hybrid of category I and II. These devices may have relatively stable sleep schedule but this sleep schedule can start changing quite often upon detection of some event. For example, a sensor device in a water dam may provide depth information of water once every few hours. It may follow a relatively stable sleep / active schedule. If water goes above a threshold, it may start providing depth information quite frequently and this frequency to increase as water level rises in that dam.

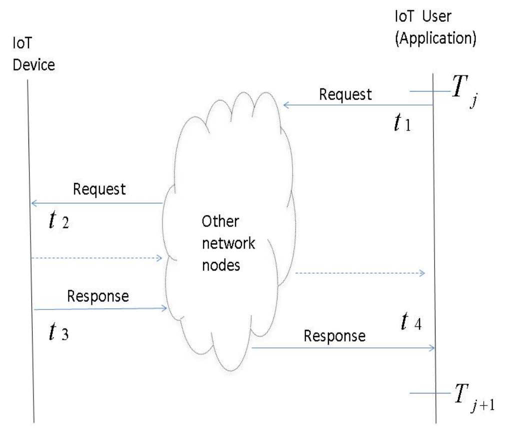
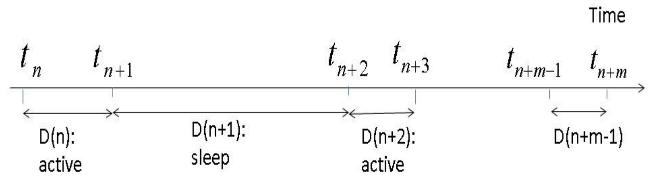


Figure 4: Delay Tolerant Application

Sleep schedule of a device is shown in Figure 5. Here, “m” is the number of time intervals that are included in this sleep schedule, “state” of the device is either active or sleep, and “bit_map_state” is a compact (array) representation of state of the device with active state represented as bit ‘1’ and sleep state as bit ‘0’. We use an IoT application (or lower) layer protocol to communicate sleep schedule from one node to another node. Constrained Application Protocol (CoAP) [9] is a lightweight application layer protocol and runs over UDP. We show one way to communicate sleep schedule using Constrained Application Protocol (CoAP) in Figure 6. Here, IoT Service Platform uses CoAP Observe mode command and asks IoT GW to convey sleep schedule of devices to it. CoAP Observe mode is used for asynchronous communication and it also allows GW to send notifications to IoT Service Platform whenever sleep schedule of an associated device changes. This works well for the cases where IoT GW decides sleep schedule of devices or has information about it using lower layer mechanisms. This may not be always true as devices may control their own sleep schedule in some scenarios. As shown in the same figure, this CoAP Observe command can also be used by the GW to get sleep schedule from the devices directly. It is also possible to optimize this protocol sequence by adding a CoAP option packet where “option” can represent sleep schedule and this option packet can be piggybacked on other CoAP messages.

A. Application Assisted Power Saving for Devices

To conserve power of devices, we want to keep these devices in sleep mode as long as possible. As in [10], we publish state of devices in a device directory. This state could be published by the corresponding GW or could be directly published by the devices. Once an IoT application generates request for a device j, it first checks state of the device j in the device directory. If device j is in sleep state and if it is a delay tolerant request, we buffer the request at the application (or at IoT service platform). Alternatively, the application can delay generating the request if feasible. Goal is to delay sending request to device if device is in sleep state without violating delay requirements of the IoT request. This step is shown in Figure 7.



$$t_{n+k} = t_{n+k-1} + \delta_{n+k-1}, 1 \le \mathrm{k} \le \mathrm{m}$$

$$sleep\_sched : [m; (D(n), active),$$

$$(D(n+1), sleep), \ldots\ldots, (D(n+m-1), state)]$$

$$\text{Compact representation :}$$

$$[m; t_n; \delta_\mathrm{n}, \delta_{n+1}, \ldots\ldots, \delta_{n+m-1}; bit\_map\_state[m]]$$

Figure 5: Active / Sleep Schedule for a device

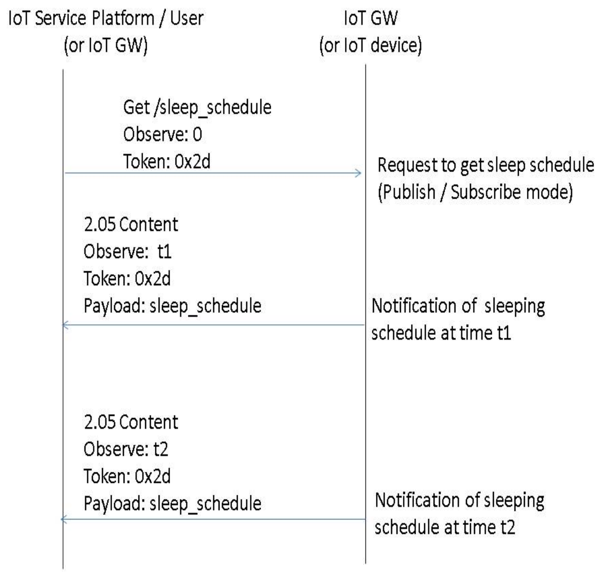

IoT Service Platform / User (or IoT GW)
IoT GW (or IoT device)
Get /sleep_schedule
Observe: 0
Token: 0x2d
Request to get sleep schedule (Publish / Subscribe mode)
2.05 Content
Observe: t1
Token: 0x2d
Payload: sleep_schedule
Notification of sleeping schedule at time t1
2.05 Content
Observe: t2
Token: 0x2d
Payload: sleep_schedule
Notification of sleeping schedule at time t2

Figure 6: Use of CoAP to communicate sleep schedule

We consider a scenario where this IoT data is buffered at an IoT service platform. Let’s say that IoT request j arrives at the

IoT service platform at time $t^{req}(j)$ and response is needed by time $t^{resp}(j)$. As this request arrives at the service platform, corresponding device could be in sleep or active state. We say that $t^{first}(j)$ is the first instant, $t^{poss}(j)$ is a possible time instant and $t^{last}(j)$ is the last time instant when we want to send IoT request j from the IoT service platform to the target device. We choose these time instants so that there is some chance that request may arrive at the device when it is in active state. We also assume here that D(h) ≥ T for all h where state of device is in active state during D(h). This assumption can be easily relaxed. Here, T = round trip time in the network + β, where β≥0, is a tolerance factor which could be pre-configured or computed dynamically using some policies. This factor, β, is also bounded by a pre-configured threshold. Note that we propose a heuristic here as finding the exact optimal solution (in real-time) is computationally hard problem. We show a subset of possibilities here.

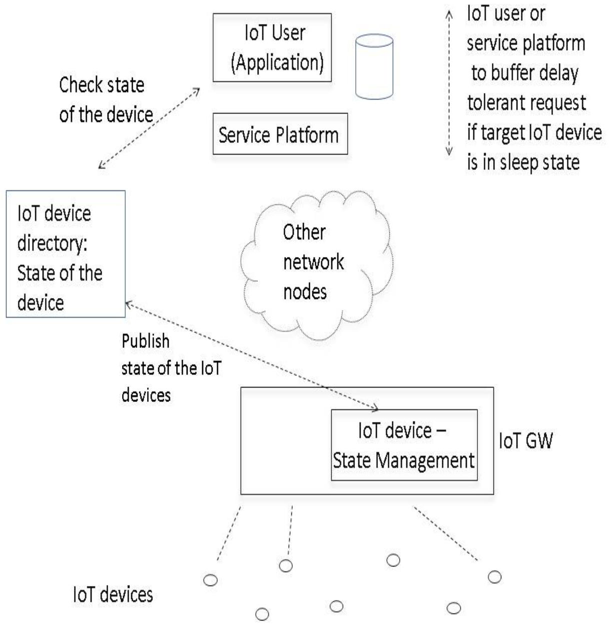

IoT User (Application)
Service Platform
Check state of the device
IoT user or service platform to buffer delay tolerant request if target IoT device is in sleep state
IoT device directory: State of the device
Other network nodes
Publish state of the IoT devices
IoT device – State Management
IoT GW
IoT devices

Figure 7: Application (User) Assisted Power saving for Devices

Let's assume that $t^{req} \in D(n-1)$, $t^{resp}(j) \in D(n+x)$, $t^{req}(j) \le t^{first}(j) \le t^{poss}(j) \le t^{last}(j) \le t^{resp}(j)$, for each IoT request j

Last time instant, $t^{last}(j)$, when we may want to transmit this request from the IoT service platform:

$$t^{last}(j) = \begin{cases} t^{resp}(j) - T, \text{if } D(n+x)\text{:}\mathit{active}, \\ t_{n+x} - T \text{ if } D(n+x) : \mathit{sleep} \end{cases}$$

First time instant, $t^{first}(j)$, when we may want to transmit this request from the IoT service platform:

$$t^{first}(j) = \begin{cases} \min(t^{req}(t), t_n - T), \mathit{if}\ D(n-1) : \mathit{active} \\ \qquad \mathit{and}\ t_n - t^{req} \ge T; \\ t_{n+1} - T, \mathit{if}\ D(n\text{-}1)\text{:}\mathit{active} \\ \qquad \mathit{and}\ t_n - t^{req} < T; \\ \max((t_n - T), t^{req}(j)) \\ \qquad \mathit{if}\, D(n-1)\text{:}\mathit{sleep} \end{cases}$$

Possible time instants when we may want to transmit this request from the IoT service platform to the corresponding device:

$$t^{poss}(j) \in \begin{cases} (t_{n+a+1} - T, t_{n+a+1})\ \mathit{for}\ D(n+a) : \mathit{sleep}, \\ \qquad t^{poss}(j) \in D(n+a), 0 \le a \le x-2; \\ (t_{n+a}, t_{n+a+1} - T)\ \mathit{for}\ D(n+a) : \mathit{active}, \\ \qquad t^{poss}(j) \in D(n+a), 0 \le a \le x-2; \\ (t^{first}, t_n),\ \mathit{for}\ D(n-1) : \mathit{sleep\ and} \\ \qquad t^{poss}(j) \in D(n-1); \\ (t^{first}, t_n - T)\ \mathit{for}\ D(n-1) : \mathit{active}, \\ \qquad \mathrm{t_n - t^{req} \ge T},\ t^{poss}(j) \in D(n-1); \\ (t_{n+x-1}, t^{last})\ \mathit{for}\ D(n+x) : \mathit{sleep\ and} \\ \qquad t^{poss}(j) \in D(n+x\text{-}1); \\ (t_{n+x}\text{-}T, t^{resp} - T)\ \mathit{for}\ D(n+x)\text{:}\mathit{active\ and} \\ \qquad t^{poss}(j) \in D(n+x-1)\ \mathit{or} \\ \qquad t^{poss}(j) \in D(n+x) \end{cases}$$

One heuristic technique would be to transmit this request at the last possible time instant, $t^{last}(j)$, from the IoT service platform. We can also look at other factors such as congestion indicators and choose one of the possible time instants, $t^{poss}(j)$, when there is no (or low) congestion in the network.

B. Network Assisted Power Saving for Devices

It may not be always possible to use application assisted power saving scheme for devices. This could happen due to variety of reasons. An application or service platform may not have required buffering capability to handle large number of IoT requests. It may not have access to a device directory where it can check current state of the device. Some of the devices may change their state quite frequently and state of such devices may not even be available in the device directory. An IoT request sent by an IoT application or service platform may arrive at an IoT GW but some of the wireless interfaces of the GW may not be available at that time. This could happen due to congestion in the corresponding wireless network or it could be that the GW has moved some of those interfaces in "sleep" state to conserve power. In the framework proposed here, we also handle the scenarios where IoT application assisted methods may have such limitations and one such method is specified in this sub-section.

If a device is in sleep state and its associated IoT GW has received a message for it, GW sends special message (such as beacons or paging message as in Figure 9) to indicate to device that it has data pending for that device. This allows device to be in sleep state and only wake up at specific time instants to check if there is any packet pending for this device at the IoT GW (or the corresponding wireless AP or wireless coordinator if that is part of the GW). If there is data pending for this device (as indicated by beacon message), device sends data request to the corresponding GW and gets the data frame. Device processes this data payload and provides an IoT Response as shown in Figure 9.

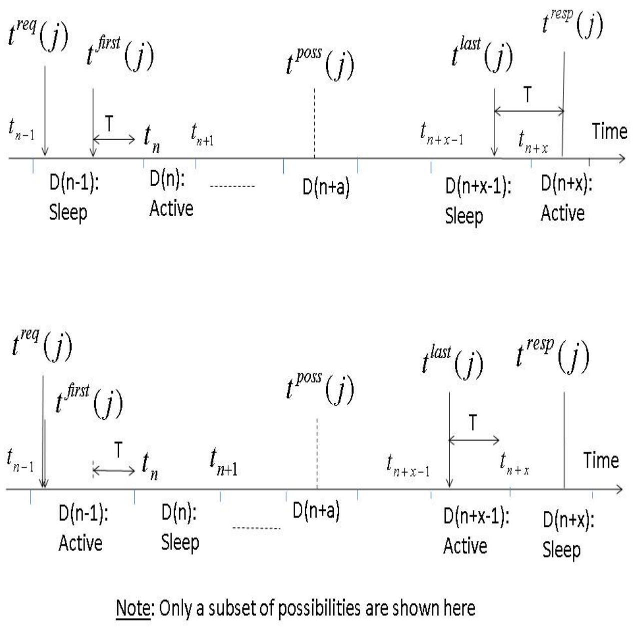


Figure 8: First, last and possible time instants when we may want to transmit IoT request j from IoT service platform to the associated device

We show our enhanced IoT request / response operation for an IEEE 802.15.4 type of network in Figure 10. As shown in this figure, GW receives IoT Request data from IoT application (or service platform) for a device that is in sleep mode. Instead of indicating this to the device in the next beacon, GW decides to buffer this IoT Request and indicates "pending data" to the corresponding device after m beacons where m is computed considering various factors such as delay constraints of the request, remaining battery life of the device (if known), congestion in the network and other QoS factors. We propose to use machine learning algorithms to predict number of additional requests that may be received for this device while one of its requests is pending at the GW. It would help to intelligently aggregate these IoT requests for this device and help to keep the device in the sleep state as long as possible.

C. Device & Network Assisted Power Saving for Devices

If a device needs to send data to IoT service platform or a user, it can consider several factors such as delay requirement of that data, power level of the device, state of the GW if known, and decide when to transmit data. If a device receives a beacon from an IoT GW indicating that there is data pending for this device, this device is usually supposed to contact IoT GW and get data that is pending for it. In our framework, we allow a device to decide whether or not it immediately wants to respond to the beacon message where it is indicated that there is data pending for this device. As shown in Figure 11, the device can decide to wait for some time before deciding to respond to such beacon messages. If a device doesn't respond to such beacon message, the GW indicates "data pending" in each of the next beacon message until it reaches a maximum retry limit. We also allow the GW to indicate "pending data" to device only at select beacons instants and do not make it necessary for it to inform about pending data on each and every beacon until it gets response from the device. This scheme also allows device to be in sleep mode for longer duration and helps to conserve its power.

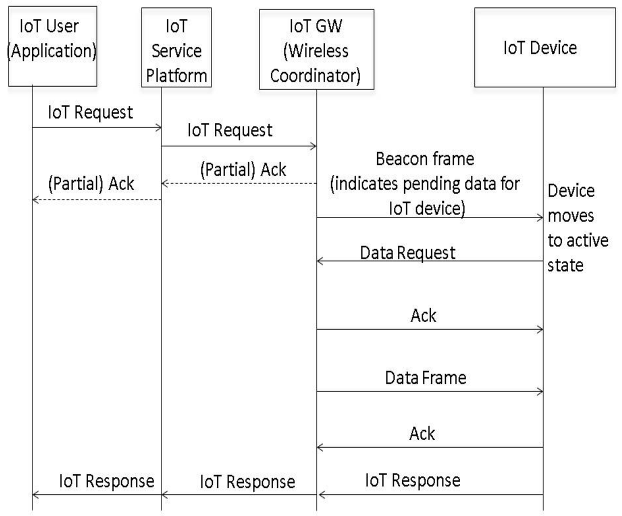


Figure 9: IoT Request / Response – Normal operation

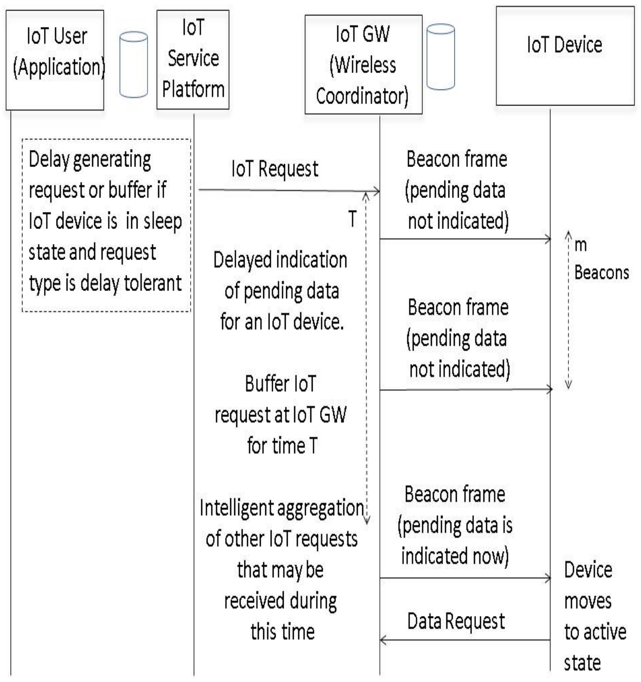

IoT User (Application)
IoT Service Platform
IoT GW (Wireless Coordinator)
IoT Device
Delay generating request or buffer if IoT device is in sleep state and request type is delay tolerant
IoT Request
T
Delayed indication of pending data for an IoT device.
Buffer IoT request at IoT GW for time T
Intelligent aggregation of other IoT requests that may be received during this time
Beacon frame (pending data not indicated)
Beacon frame (pending data not indicated)
Beacon frame (pending data is indicated now)
Data Request
m Beacons
Device moves to active state

Figure 10: Delayed indication of pending data by IoT GW

III. POWER SAVING FOR IOT GW

A generic IoT GW with different types of interfaces is shown in Figure 3. It has compute modules that do the computation part. For example, we may do local data analytics at this GW for data received from devices. In addition, it could have a subset of communication modules such as WiFi-Direct, LTE-Direct, Cellular AP, WiFi AP, Cellular / WiFi UE and IEEE802.15.4. It could also have Ethernet and Serial interfaces. To reduce power consumption of GW, we want to keep some of these communication modules in the sleep state as long as possible.

We use a module called "Communication Management" as shown in Figure 12. As data arrives at an IoT GW that needs to be sent to a device, it checks QoS requirements of the data, sleep / active state of the corresponding IEEE 802.15.4 coordinator (which is part of this IoT GW) and state of the device. If 802.15.4 coordinator is in sleep mode, we buffer this data and allow the interface for the 802.15.4 coordinator to be in sleep mode as long as possible. Other requests destined for the same coordinator may also arrive at the GW during this period. This allows the GW to analyze these multiple requests and intelligently aggregate these before sending to the corresponding device. One possible way to know about request pattern is to learn from previous request patterns and use machine learning algorithms to predict possible future patterns.

We show our scheme for LTE type of networks in Figure 13. Here, LTE UE is part of IoT GW. We reduce power consumption of IoT GW by keeping this UE in sleep (idle) mode by buffering and delaying paging message at Serving Gateway (S-GW) in the network. We allow computation of this buffering duration at the S-GW using machine learning algorithms where S-GW can use previous requests patterns to predict future request patterns and it can estimate the duration to buffer this request. For computing this duration, it can consider factors such as delay requirements of the IoT request, number of additional messages that may arrive at the S-GW during this period and congestion indicators from different network nodes. As S-GW buffers this initial request, it can intelligently aggregate data from multiple messages for that device if additional requests are received during this time period. S-GW provides paging message to Mobility Management Entity (MME) after this computed duration and MME transmits this to LTE base stations (eNodeBs) that are in the tracking area list for that UE.

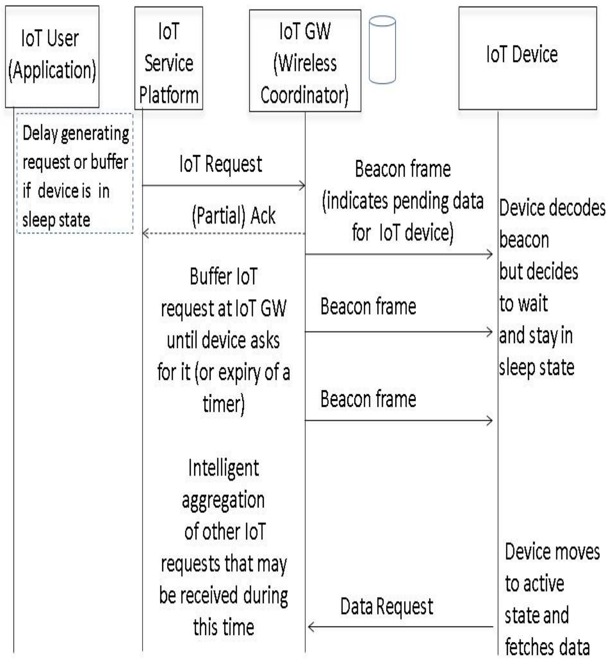

IoT User (Application)
IoT Service Platform
IoT GW (Wireless Coordinator)
IoT Device
Delay generating request or buffer if device is in sleep state
IoT Request
(Partial) Ack
Buffer IoT request at IoT GW until device asks for it (or expiry of a timer)
Intelligent aggregation of other IoT requests that may be received during this time
Beacon frame (indicates pending data for IoT device)
Beacon frame
Beacon frame
Data Request
Device decodes beacon but decides to wait and stay in sleep state
Device moves to active state and fetches data

Figure 11: Delayed data fetching by device

IV. CONCLUSIONS AND FUTURE WORK

We have presented a framework to reduce power consumption of IoT / M2M devices and gateway nodes in this paper. It should be noted that none of these methods introduce any new synchronization requirements for the wireless access technology used. IEEE 802.15.4 has existing methods to synchronize devices with time slots used by the 802.15.4 coordinator. These methods work at RF, Physical and MAC layers, and we continue using those methods without any changes here. Similarly, LTE networks have lower layer methods to synchronize LTE devices with the LTE base station. We continue using those methods as well. Our specified methods work at higher layers though they impact data delivery that is to be carried out lower layers. We also do not really introduce any new signaling overhead in most of these methods. In some methods, we buffer data for certain duration and delay indicating about "buffered data" to device. It does not add any new signaling overhead. In the device assisted method (as shown in Figure 11), we do introduce limited signaling overhead as GW may need to indicate "pending data" to corresponding device in multiple beacon periods if device doesn't respond immediately. But, this overhead can be reduced as device can decide to respond quickly or GW need not indicate "pending data" with each and every beacon (i.e. GW can skip indicating "pending data" at certain beacon instants). Tradeoff between power saving and signaling overhead for this particular method needs to be studied further. Note that we do this in a way that response time (or delay) constraints are not violated.

We have intuitively explained that these methods potentially help to reduce power consumption of devices and gateway nodes by keeping them in sleep state for longer duration and by intelligently aggregating messages at the IoT GW. We have also presented software building blocks of this framework. Future work includes simulations of the mechanisms presented here.

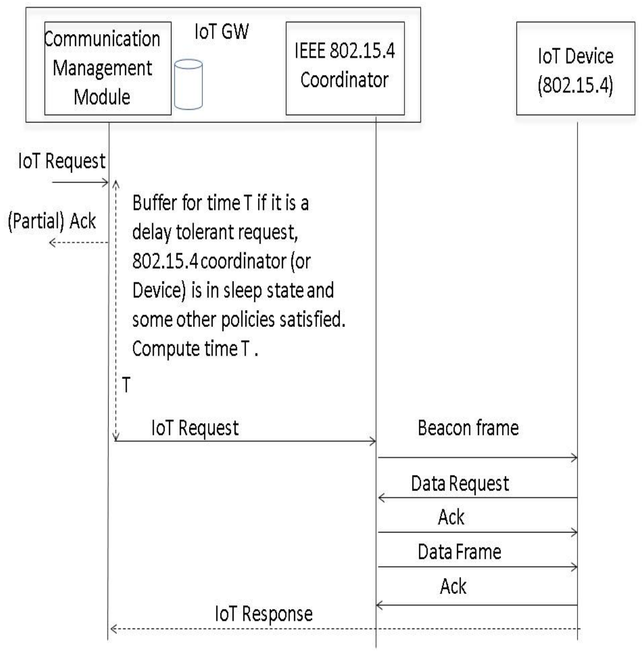

Communication Management Module
IoT GW
IEEE 802.15.4 Coordinator
IoT Device (802.15.4)
IoT Request
(Partial) Ack
Buffer for time T if it is a delay tolerant request, 802.15.4 coordinator (or Device) is in sleep state and some other policies satisfied. Compute time T .
T
IoT Request
Beacon frame
Data Request
Ack
Data Frame
Ack
IoT Response

Figure 12: Delayed data indication within an IoT GW

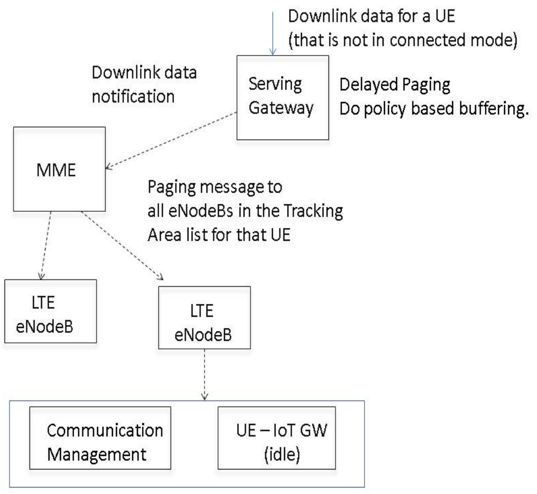

Downlink data for a UE (that is not in connected mode)
Downlink data notification
Serving Gateway
Delayed Paging
Do policy based buffering.
MME
Paging message to all eNodeBs in the Tracking Area list for that UE
LTE eNodeB
LTE eNodeB
Communication Management
UE – IoT GW (idle)

Figure 13: Delayed Paging for LTE Network

REFERENCES

[1] Machine to Machine Communications (M2M): M2M service Requirements, ETSI TS 102.689 V1.1.1, 2010-08.

[2] Jia-Ming Liang, Jen-Jee Chen, Hung-Hsin Cheng and Yu-Chee Tseng, "An Energy Efficient Sleep Scheduling with QoS Consideration in 3GPP LTE-Advanced Networks for Internet of Things," IEEE Journal of Emerging and Selected Topics in Circuits and Systems, Volume 3, Number 1, March 2013.

[3] H. Chao, Y. Chen and J. Wu, "Power Saving for Machine to Machine Communication in Cellular Networks" International Workshop on Machine-to-Machine Communications, Globecom Workshops, 2011, pp. 389-393.

[4] Hsing-Ho Lin, Hung-Yu Weim and Rath Vannithamby, "Deep Sleep: IEEE802.11 Enhancements for Energy Harvesting Machine-to-Machine Communications," Wireless Networking Symposium, Globecom 2012, pp. 5231-5236.

[5] Low-Rate Wireless Personal Area Networks (LR-WPANs), Amendment 1: MAC Sublayer, IEEE 802.15.4e-2012.

[6] M. Taneja, "Lightweight Security Protocols for Smart Metering," IEEE Innovative Smart Grid Technologies (ISGT), November 2013, pp. 1 -5.

[7] R. P. Liu, G. J. Sutton and I. B. Collings, "WLAN Power Save with Offset Listen Interval for Machine to Machine Communications," IEEE Transactions on Wireless Communications, Issue 99, 2014, pp. 1-11.

[8] S. C. Jha, A. T. Koc, M. Gupta and R. Vannithamby, "Power Saving Mechanisms for M2M Communication over LTE Networks," 2013 Forst International Black Sea Conference on Communication and Networking, pp. 102-106.

[9] Z. Shelby, K. Hartke and C. Bormann, "Constrained Application Protocol (CoAP)," IETF draft-ietf-core-coap-18 .

[10] Z. Shelby, C. Bormann, S. Kcro, "CoRE (Constrained RESTful Environment) Resource Directory," IETF draft-ietf-core-resource-directory-01