**How oneM2M is Enabling Sustainable IoT Deployments**

Today, IoT devices already outnumber smartphones 2 to 1 and this gap is expected to widen to 4 to 1 by [2025](https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/#:~:text=The%20total%20installed%20base%20of,that%20are%20expected%20in%202021) [1]. Due to their growing numbers, optimizing power consumption and carbon footprint of IoT devices and networks has been a top priority for oneM2M. oneM2M, a global IoT standards body specifically targeting the standardization of a common service layer for IoT devices and networks, has a dedicated sustainability committee focusing on how IoT devices and networks can be deployed and maintained in a more sustainable manner. As part of its charter, the oneM2M Sustainability Committee has studied sustainability challenges for IoT deployments and identified several oneM2M defined features which can be used to address these challenges.

**IoT Sustainability Challenges**

*IoT Devices*

IoT devices differ from smart phones in a few key ways. IoT devices typically handle comparatively small amounts of data and are instead optimized to prioritize battery life. Battery-powered IoT devices are required to remain operational for long durations (e.g. a decade or more) without recharging or replacement of batteries. These devices often have a much smaller battery or no battery at all and instead rely on harvesting ambient energy from sources such as solar or vibrations. As such, to conserve their energy, many of these devices need to remain disconnected and completely powered off for long durations until they are awoken to perform an operation, for instance a sensor that wakes-up monthly to report a reading and then returns to sleep. Other devices must minimize the amount of circuitry they leave powered on so they are still able to be remotely triggered to fully power-up if/when needed, for example an actuator that must be sent a remote command to close a critical valve in a timely manner. Unlike smartphones which can rely on users for daily recharging, and/or transitioning in and out of low power modes when needed (e.g., switch to airplane mode, or dim the device display), IoT devices typically cannot rely on user interaction. Instead, these connected IoT devices must rely on more autonomous forms of management and control within the devices themselves or the networks to which they connect to help them conserve energy.

*IoT Networks*

Networks supporting IoT devices typically consist of a diverse collection of IoT devices as well as non-IoT devices. These devices often have different and even conflicting requirements such as network bandwidth, latency and reliability. Balancing and meeting the requirements of these different types of devices can be challenging for networks let alone doing so in a sustainable manner. To deploy networks in a sustainable manner, networks protocols and the usage of the network by devices must be optimized. For example, protocols must be optimized to minimize unnecessary overhead within individual messages as well as in the message exchanges that occur over the network. In addition, scheduling of communication to reduce the peak load on the network is also very important. These optimizations can reduce amount of network resources (e.g., servers, switches, routers) required to reliable service network traffic. This in turn can reduce the energy and carbon footprint of the network and therefore enhance its sustainability.

*IoT Systems*

IoT systems consist of a network of networks with each network typically consisting of different types of devices, protocols, services and applications. These differences pose sustainability challenges for IoT systems. For example, the lifetime of an IoT system is often dictated by how well the system can adapt and interwork diverse networks and devices together with one another. When older networks and devices are unable to be easily interworked with newer ones, then this typically requires swapping out older technologies and replacing them with newer ones. This cannot only be costly from a financial perspective but also from a carbon footprint perspective since it is a major source of e-waste. Enabling systems with the capability to efficiently interwork older networks and devices with newer ones can prolong the lifetime of these technologies and greatly enhance the sustainability of IoT systems.

**Solutions**

oneM2M has defined several key features within the oneM2M standard that can be used to enhance the sustainability of IoT devices, networks, and systems. These features reside within the different types of services supported by the oneM2M service layer. The oneM2M service layer is typically deployed on a cloud server and depending on deployment requirements may also be deployed on an edge server or gateway. The service layer can even be deployed on IoT devices themselves if they have adequate resources to host the service layer. The features of the oneM2M service layer can help minimize the power consumption and carbon footprints of IoT devices, networks and systems and in turn enhance their sustainability. The following are some examples of these oneM2M features.

<Consider adding a figure showing oneM2M service layer used in a typical IoT deployment>

*IoT Device Twins*

Device twin technology enables physical IoT devices deployed in the field to be represented by digital counterparts that reside within services hosted on a cloud or edge server. These digital counterparts, or twins, are then used to exchange information between the devices themselves and the applications and services that wish to interact with them. For example, a sensor can publish readings to its digital device twin. Use of device twin technology can provide a huge power consumption benefit for IoT devices because device twins allow IoT devices to disconnect, power down and sleep for longer durations of time without impacting the functionality of applications, services and other devices that require data from these IoT devices. Even while IoT devices are sleeping, their data can be accessed from their device twin.

Device twin technology also reduces the number of requests IoT devices must send and receive, since they only need to interact directly with their device twins instead of other applications, services, or devices. An IoT device is only required to wake-up long enough to send an update to its device twin, and then can immediately go back to sleep. Even an actuator-type device, which requires receiving commands, can benefit from device twins because actuator commands can be stored within the digital twin. The IoT device can then wake-up based on its own schedule, check its device twin for any new commands, perform these commands, update its device twin with the results, and then return to sleep.

The oneM2M standard defines several different types of device twin resources providing developers with flexible options for representing their devices and applications as digital counterparts in the oneM2M service layer. Based on use case requirements, developers can choose different types of digital twin resources (e.g., AE, container or flexContainer resources to name just a few).

*IoT Message Profiles*

Many IoT devices generate a stream of sensor readings on a periodic schedule or event basis. The number of bytes of data included within each message the device sends or receives can have a major impact on the battery life of the device. The larger the message sent or received over the network, the longer that device must remain powered to send the message. Over time, this can add up and dramatically reduce the lifetime of a device’s battery.

In many IoT use cases, there is a certain amount of information contained within these messages that is static in nature and does not change. For example, certain fields within upper layer protocol headers of the message or in the data payload of the message. This static information is critical to processing the message and provides critical context to the services and applications (e.g., analytics) consuming and processing the message. However, this static information can result in significant overhead on the device and the network if included in every message the device sends. To alleviate this, oneM2M message profiles can be used. Within a oneM2M message profile, static information elements can be defined along with criteria defining which device(s) and/or message type(s) a profile is applicable to. Message profiles can be configured within oneM2M services in the network (e.g., cloud or edge) to which an IoT device sends it messages. When the oneM2M service receives messages, it compares the messages against the profiles and if a match is found, the messages are enriched with the static information defined in the profiles and further processed by the oneM2M service or applications (e.g., analytics operations are performed). This allows devices to send only the bare minimum amount of information that has dynamically changed per message (e.g., a sensor reading) and streamline the size of the message. oneM2M message profiles can also be configured on IoT devices themselves and used in a similar manner by the devices to enrich the messages they receive with static information. Therefore, the use of oneM2M message profile technology helps minimize the size of messages flowing between IoT devices and oneM2M services in the network, which in turn translates into less energy consumed by the IoT devices as well as reduced load and energy consumption by the network infrastructure used by the IoT devices.

*IoT Device Triggering*

oneM2M device triggering enables IoT devices to “sleep” for long periods of time when not in use, and intelligently “wake up” when needed. Specifically, IoT devices register to a oneM2M service in the network to provide instructions for triggering the device. The instructions can include a device’s contact information as well as its sleep schedule. Once registered, the IoT device powers down to sleep and conserve power. Based on its sleep schedule, the device can periodically wakeup for a short duration of time to listen for triggers by optimally enabling only a small, select portion of its circuitry required to receive a trigger, but otherwise, the device consumes a relatively small amount of power. If the service in the network needs to communicate with the device (e.g., send it a command), it schedules a trigger to be sent to the device via the underlying network (e.g., 3GPP cellular network) during a scheduled wake up time. If the device receives a trigger, it fully powers-up, connects to the network and communicates with, or receives a command from, the service. Afterwards, the device powers back down and goes back to sleep to continue conserving its power.

*IoT Event Processing*

Offloading event detection and processing from user applications to oneM2M services deployed in the cloud and on edge compute nodes in the network reduces the overall number of messages exchanged over the network. Rather than an application retrieving every sensor reading published by the device(s) it is interested in and checking whether sensor readings have crossed a certain threshold value of interest, the application instead offloads an event detection rule to a oneM2M service in the network. Within the rule, the application specifies the devices it is interested in, the data of interest from these devices, conditions of interest for this data, and actions that are to be performed if/when these conditions are detected. For example, if pressure within a specified boiler device exceeds a specified threshold, then power to the boiler is turned off. Using the rule, the oneM2M service can efficiently offload and perform this work on behalf the application.

Offloading eliminates the need to send individual copies of device data to each application to process, which greatly reduces the amount of messaging in the system. It also enables the pooling and reuse of compute and storage resources of the oneM2M service by the IoT devices and applications in the system. For example, data offloaded by IoT devices to a oneM2M service in the network can be used both for the monitoring of a single condition defined in a single rule from a single application, and also simultaneously used by the oneM2M service to monitor many conditions defined in many rules from many applications. As result, offloading IoT event processing can play a major role in helping optimize energy consumption in the overall 5G system.

*Time Synchronization & Compensation*

Unlike non-resource constrained device which have the capability to keep their local time synchronized with other devices in the network using technologies such as GPS and Network Time Protocol (NTP), many IoT devices lack this capability due to its overhead. The inability for IoT devices to keep their local time synchronized with other devices, services and applications in the network can have catastrophic effects. For example, if a patient’s medical device timestamps a sensor reading with a date and time which is not synchronized with the medical staff that is analyzing the readings, then they may mis diagnose and treat a patient.

The oneM2M time management service supports a set of time management capabilities to minimize time synchronization overhead on IoT devices allowing these devices to remain synchronized with the rest of the system without introducing a lot of extra overhead. For example, the oneM2M time management service supports the capability to perform time compensation on behalf of IoT devices by adjusting time related metadata contained in the messages sent by a device to the oneM2M service. The oneM2M time management service is able to monitor and detect time synchronization offsets of IoT devices and the rest of the system. Based on the detected offsets, the oneM2M service is able to compensate for the time offsets on behalf of the devices. For example, when messages are received from devices which contain timestamp information, the oneM2M service can adjust the timestamps to compensate for any detected time offsets. This alleviates the devices from the burden and overhead of having to maintain time synchronization with the rest of the system.

*Load Balancing Requests*

For certain use cases such as smart agriculture and environmental sensing deploying the same type of IoT sensor device in large numbers in the same geographical region is common. For example, to monitor soil moisture in a field or air quality in forest region several sensors may be deployed. This not only adds redundancy in case of device failure but can also be used to extend the overall deployment lifetime of the devices if proper load balancing is performed across the devices. oneM2M supports the capability to group IoT devices together such that requests can be load balanced across the group. When a oneM2M service receives a request targeting a load balancing group, a device can be selected from the group to receive the request. The selection of the device can be made based on a configured algorithm such as the least loaded device (e.g., device with highest battery level), round-robin, weighted round-robin, etc. The oneM2M service then sends the request to the selected device but not the other devices in the group. In doing so, the oneM2M service intelligently load balances requests across the group of devices and prolongs the deployment lifetime of the group of devices such that the group of devices can monitor a given region for a more extended period of time than a single device could (at the same measurement sampling rate).

*IoT Message Delivery Scheduling and Aggregation*

<Add Description>

*Offloading Time Series Processing*

<Add Description>

*Network Congestion Detecting and Message Throttling*

<Add Description>

Smoothing out demand on a network such that peak loading is reduced on the network is off can help reduce the amount of network Throttling Messages to help alleviate congestionTBD

* *Assist with managing congestion in underlying network transports [<nwMonitoringRequest>]*

*<Consider one or more of the following as possible additions>*

* *Offloading a software management campaign targeting a set of devices from an application to service layer [<softwareCampaign>, <mgmtObj>]*
* *Interworking to different IoT technologies [IPEs]*
* *Managing end-to-end sessions between applications [<e2eQosSession>]*
* *Offloading semantic reasoning operations from applications to service layer [<reasoningRules>, <reasoningJobInstance>]*
* *Offloading and managing data storage and brokering on behalf of devices and applications (e.g., storage, discovery, authorization, stats collection, charging of data) [data management resources, <accessControlPolicy>, <statsConfig>, <eventConfig>, <statsCollect>]*
* *Offloading semantic mashup operations from applications to service layer [<semanticMashupJobProfile>, <semanticMashupInstance>, <semanticMashupResult>]*
* *Sending requests to 3GPP devices using Non-IP Data Delivery (use of 3GPP control plane to deliver messages)*
* *Managing network connectivity used to communicate with devices [CMDH]*

**Conclusion**

As the mass scale out of connected IoT devices, networks and systems continues, the importance of using and responsibly deploying IoT technologies such as the oneM2M standardized service layer described above will become critical. This will help reduce the power consumption and carbon footprints and make these deployments more sustainable. In addition to the features described above, the oneM2M service layer supports several additional features described in the oneM2M specifications that are publicly available at [www.oneM2M.org](http://www.oneM2M.org). oneM2M is also committed and actively engaged in identifying additional features that can further optimize the sustainability of IoT deployments and encourages and welcomes new participants in this effort.

[1] <https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/#:~:text=The%20total%20installed%20base%20of,that%20are%20expected%20in%202021>.