What is IoT?

Understanding IoT, Devices and Standards

Table of contents

Table of figures

1. Introduction

Internet of Things (IoT) is a broad term, and everyone seems to have a different notion of what it is.

This paper offers Friendly Technologies' view of the IoT Device Management's fundamental concepts. The paper also presents some of the current leading IoT technologies and reviews a few management protocols and their use in Device Management and IoT.

After reading this white paper, you should have a deeper understanding of the IoT, possess greater familiarity with standard IoT protocols and clients, and be able to choose the appropriate type of client for specific IoT scenarios.

2. IoT and Device Management

This part of the paper describes various IoT devices, platforms and IoT installation scenarios and explains the need of device management for large-scale installations.

2.1 What are the "Things" in the Internet of Things?

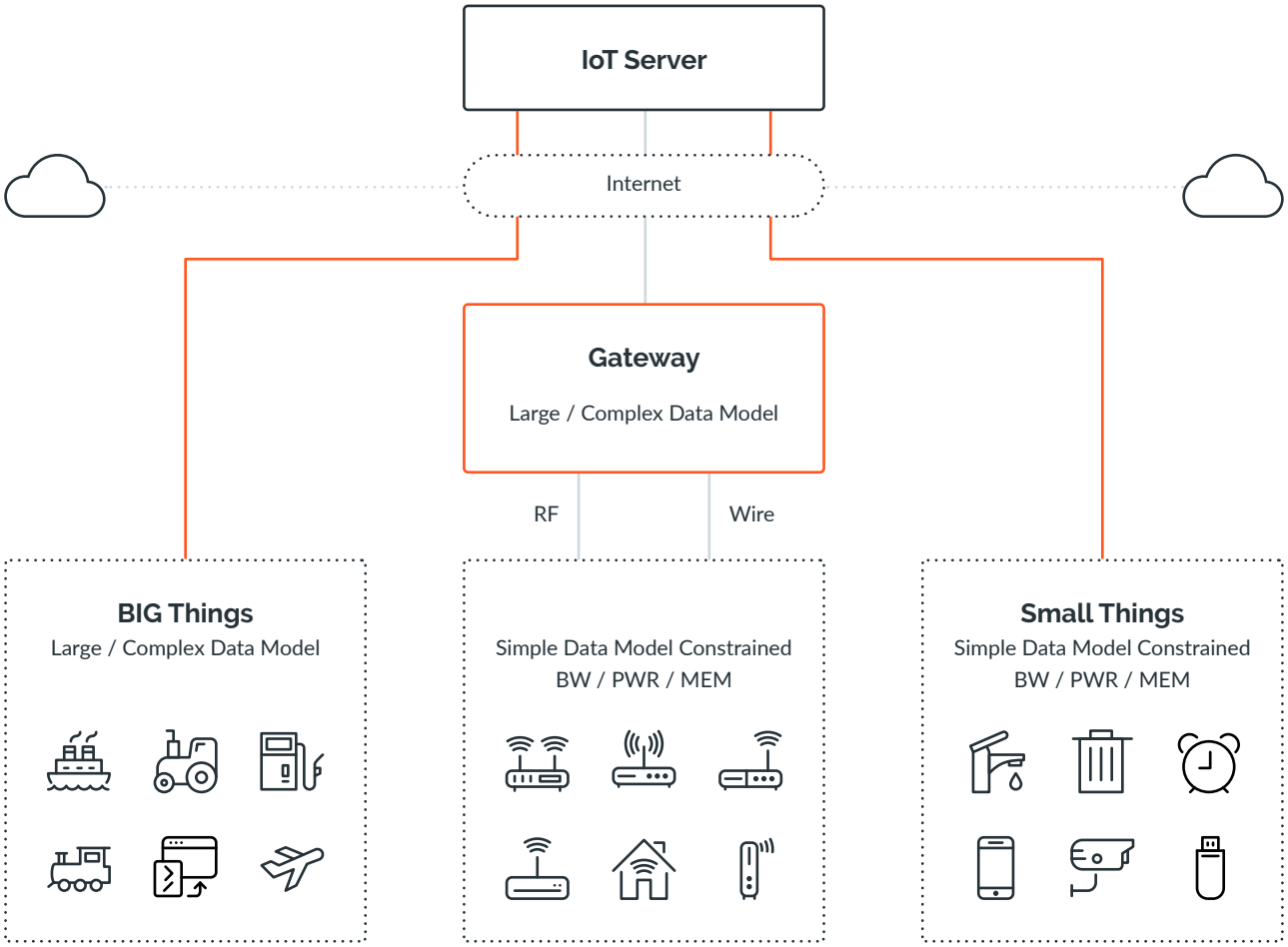

Figure 1: Standard IoT Platform

When people talk about Internet of THINGS, they may be referring to a variety of different concepts. Taking a closer look, we can find that in fact, most IoT architectures fall under one of the following three categories:

- **Big Things:** Regardless of their physical size, this category refers to the more complex devices. For example, modems, routers or industrial machines, all of which share the fact that they are generally not restricted by their power supply, are connected directly to Internet with no bandwidth constraints, and most importantly have a data model that is complex, with thousands of parameters that need to be managed.

- **Small Things:** The new category of "smart" things – simpler devices that are directly connected to the Internet – such as sensors, smart locks, smart lights, etc. Items in this category typically have a relatively limited memory and are often powered by a battery which means they also have power restrictions. In addition, they are often connected to the Internet over a SIM card and the bandwidth is either expensive or limited. These kinds of devices generally have a much simpler data model, and their functionality is often concerned mostly with the "Set" and "Get" commands. An example of a "Small Thing" is a smart light bulb, which can be turned ON or OFF, maybe dimmed or change its color, and in more sophisticated devices even report its power consumption – however, all these functionalities amount to less than 10 parameters to be controlled.

- **Small Things Connected to the Internet via a Gateway:** This category is probably going to dominate the IoT world in the not-so-distant future. One can already see it happening today. These kinds of items and devices are referred to as IoT devices, but are not really IoT because they do not have the "I" part in them; they are not Internet connected. This includes devices that communicate over protocols like ZigBee, Z-Wave, LoRa, Sigfox and BLE. While these protocols are generally considered to be IoT protocols, they do not actually work over an Internet connection. Making these into Internet capable devices requires the use of a **gateway** in the middle.

2.2 The Gateway Infrastructure

Most people today consider a gateway as simply a relay between non-IP protocols and an IP protocol. However, gateways play a much more fundamental role when it comes to large-scale IoT installations, where gateways become a more critical entity that needs a special consideration. When it comes to a large-scale IoT installation, especially one where IoT is offered by a service provider as an infrastructure, there is a need to manage a great number of gateways. While the end devices are small and simple, the gateway itself is a complex device, and understanding the complexities of IoT involves understanding the fundamental difference between a data IoT platform and an IoT and Device Management Platform.

2.3 Data centric IoT Platforms

Most vendors of IoT platforms typically provide an Internet of the data of Things Platform, with no attention to the Things themselves. These platforms have the capability to perform "Set" and "Get" on the devices, collect data, view inventory, and run Big Data, dashboards and BI analysis. They often have a rules and automation engine, and allow to connect external applications to the platform. Sometimes such platforms also allow basic firmware updates for some of the devices.

2.4 IoT and Device Management Platforms

An IoT and Device Management platform offers the same functionality and capabilities of a data-centric IoT platform, with all the data collection and manipulation tools. The big advantage of an IoT and Device Management platform over ordinary IoT platforms is that it also enables Gateway Device Management. Gateways are typically quite complex; they can collect data from tens of thousands of devices. Their data model – and specifically, the amount of data that they are handling – is both complex and dynamic. Therefore, when managing an IoT gateway infrastructure, one needs the ability to run more complex tasks such as remote diagnostics and repair, as there could be many thousands of gateways installed across a state or a country – some in very remote locations, where one cannot rely on being able to send a technician for maintenance and repair of each issue. An IoT network installation should hold for dozens of years, and one needs to be able to configure it remotely, from a central location.

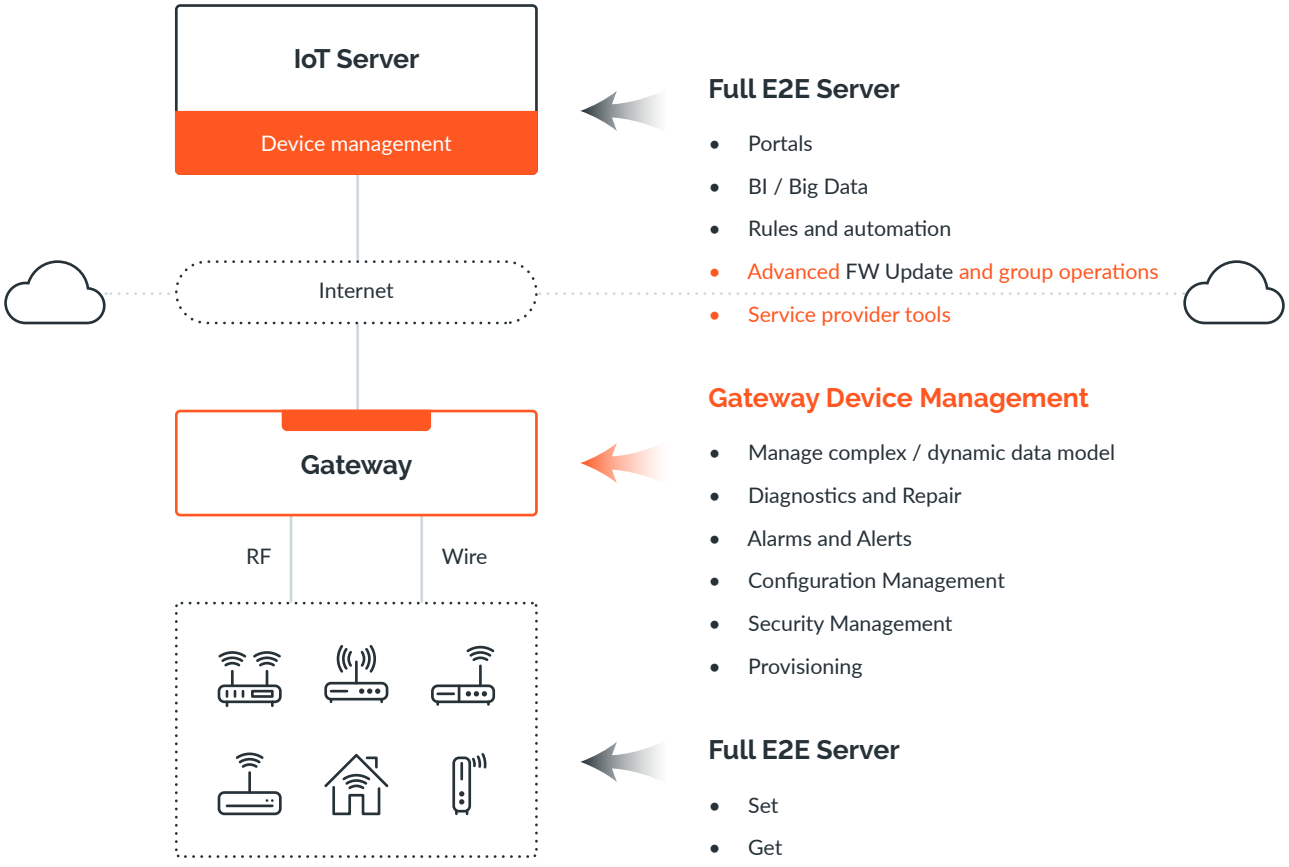

Figure 2: IoT Device Management Platform

For example, consider the case of the owner of a building with a relatively simple, smart building installation. In this case, the owner is the one who is managing the building. Typically, the owner can log into a gateway-embedded webpage and configure any parameter with ease. However, for a holding company managing hundreds of smart building, it is impossible to log into the web server of each individual gateway. It is essential to be able to run diagnostics and repair functionality, receive alarms and alerts, and run remote configuration or a mass firmware update

2.5 Security Concerns

Security is a major concern when managing gateways due to the ever-increasing number of cyber-attacks. End devices do not have a complex data model or anything to configure, and are harder to access since they do not have direct IP connectivity, so they are less likely to be hacked. It is often the gateways that are the hacker's target, since they are the first point of entry and have a direct Internet connection. If you view gateways as a simple connectivity relay and do not have access to the configuration and status of the gateway itself, you will have no way of predicting or preventing a cyber-attack.

3. Four Standard IoT Protocols

In general, there are four leading standard protocols for IoT and Device Management that are in use today:

- **MQTT:** The most prominent IoT protocol, it is in fact a messaging protocol – and as such, it is very fast and light. MQTT is a "pub-sub" protocol that works similarly to WhatsApp or chat applications. The protocol publishes a message that can be retrieved on the other end extremely quickly. This is an IoT protocol, not a Device Management protocol. It allows a user, for example, to dim a light to a required value on the fly, since it is quick enough for sensing and adjusting. It is not, however, a protocol suitable for the management of IoT devices.

- **OMA Lightweight M2M (LwM2M):** Created by the Open Mobile Alliance, it is a fast, light and structured, session-based protocol. LwM2M enables most IoT functionalities while maintaining Device Management capabilities on restricted devices.

- **OMA-DM:** A Device Management protocol created by the Open Mobile Alliance for mobile applications. This protocol is ideal for managing things that are on the move (i.e. not over a fixed IP). This is a more complex and structured protocol than LwM2M with a full Device Management functionality. It is used for telematics, mobile gateways, etc.

- **TR-369 USP:** TR-369 User Services Platform (USP) is a standard defined by the Broadband Forum, evolving from the TR-069 (CWMP). The new standard is a more flexible, secure, and standardized way to manage both existing use cases of TR-069 and new connected services such as IoT and Smart Home, at scale.

3.1 Messaging Protocols vs. Session-Based Protocols

A **messaging protocol** is a protocol that uses the Publish-Subscribe mechanism. It functions much like a bulletin board, where anybody in the network can publish a topic to the board, and anybody can subscribe to the topic. Thus, there is a disassociation of sender and receiver. Any member of a network can send and receive messages to and from each other, without needing to "know" each other. They just need to know who the "broker" is – i.e. the bulletin board.

Using a messaging protocol, devices send messages without opening session. A device simply publishes a message. In a way, it is mostly "Send and Forget" with some additional possible functionality. For example, Quality of Service and confirmation that a message is sent can be implemented. There is no visibility regarding who reads the message on the other end. The object is the topic. The topic can be "Power Switch" for a light

bulb that is in room number 3, in house number 4, with the content of the message something such as “On” or “Off.”

Messaging protocols are very light and fit battery operated, low bandwidth applications. They do not leave a large footprint for memory or storage as well. However, messaging protocols are not really protocols, but rather means of communication – much like letters in a language. To run structured commands, one needs to build out of these “letters” “words” and “sentences,” and that is when it can become more complicated. Each vendor created his own language out of this, and needs to program and invent his own protocol. When running a firmware upgrade, for example, using a messaging protocol can be quite difficult, and the result functionality will not be recognized by other devices and platforms.

Some of the common messaging protocols are XMPP, CoAP, and MQTT. In the IoT world, MQTT is currently the most mature, most widely used protocol. Platforms such as the Amazon Web Services, for example, are using MQTT, although they have built a proprietary, non-standardized language (within their “SDK”) out of these MQTT building blocks. MQTT is a standard protocol, but Device Management through MQTT is limited, complex and proprietary.

A **session-based protocol**, on the other hand, is normally very standardized. The downside of such protocols is that every transaction is defined as a session – which makes communication a bit slower. Normally, the client (i.e. the device) is initiating a session with the server. The process involves opening a session, exchanging keys, exchanging information and procedures (including verification) and ending the session. This type of interaction puts a load on the server and takes more time than sending a message. However, it provides more reliability and security and allows for greater standardization.

In a session-based protocol, there is a payload, i.e., the actual data that is being sent during the session. Depending on the protocol, the data transferred is either an XML or a SOAP message (as opposed to MQTT where you can send any type of file or text as the payload). The advantage of a session-based protocol is that it has an established business logic. When sending commands like “Firmware Update” over a session-based protocol, all devices in the network know how to understand and handle the command. The data model of devices is of a known structure and server-client communication is highly standardized. Every client can negotiate with every server and vice versa. Common objects and commands are pre-defined, so there is no need to invent “building blocks.”

Because session-based protocols have a more significant load than messaging protocols, the session-based protocols are slower. Mature platforms such the Friendly One-IoT™ Platform, designed to manage tens of millions of devices on a single platform, have the capability of handling the load efficiently.

4. Standard Protocols Comparison

This part of the paper offers a comparison of four standard protocols in terms of communication and functionality.

4.1 Communication

	MQTT	LwM2M	OMA-DM	TR-369 USP
Type	Messaging	Session	Session	Session
Overhead	-	Lighter	Heavier	Heavier
Footprint	Lighter	Lighter	Heavier	Heavier
Server Load	Lighter	Lighter	Heavier	Heavier
Data model	✗ (Unstructured)	✓ Structured	✓ Structured	✓ Highly Structured
Complex Data Model	✗ (Unstructured)	✓	✓	✓
Dynamic	✗	✓	✓	✓
FW Upgrade	Proprietary	✓	✓	✓
Dm	✗ (Proprietary)	Normal	Normal	Full
Response Time	Very Fast	Fast	Slower	Slower
Common Use	Sensors, Small / Constrained Devices	Small GW, Sensors, Small / Constrained Devices	Mobile Gateways	Fixed Gateways

Figure 3: Standard Protocols Communication Comparison

This table illustrates several key points about messaging protocols and session-based protocols:

- **Overhead:** MQTT is a messaging protocol, while the other three are session-based. A messaging protocol does not have any overhead.

- **Footprint:** LwM2M has lighter overhead than other session-based protocols. OMA-DM and the TR-369 USP have greater overhead. The footprint for MQTT is very light, and fits in up to 20K. LwM2M is somewhat heavier, but it is still less than 100K in code. OMA-DM and TR-369 USP require more memory resources, as they are designed to handle tasks that are more complex.

- **Server Load:** The server load in MQTT is virtually non-existent, because there is no real server. Rather, it acts as more of a broker. The actual IoT server sits on top of it and is a subscriber to these messages. It then converts messages that it is receiving into a structured database. LwM2M, with its lighter server load, is not a lot heavier than MQTT because the transactions are small, and the data model that is being handled is smaller. OMA-DM and TR-369 USP have greater server loads.

- **Data Model:** MQTT is a channel. It does not have a real data model, and when it is being used, a data model needs to be built on the fly. This is because MQTT is just a messaging protocol. In contrast, LwM2M, OMA-DM, and TR-369 USP have structured data models, in which the model looks like a tree: There is a topic and a sub-topic, and each one has its own parameters on which you can perform "Set" or "Get" functionalities. TR-369 USP is slightly more structured than the others. Without getting into the details of the complexity of the data model, with LwM2M one can handle dozens or up to a hundred items per data model, while with OMA-DM and TR-369 USP one can handle thousands.

- **Flexibility:** If a server requires a change in the data model – for example, opening another instance, or opening another room – in TR-369 USP, LwM2M, and OMA-DM, this can be done. The server can expand, reduce, or modify the data model. In MQTT there is no real data model.

- **Firmware upgrades:** With MQTT, it is necessary to write the firmware update process from scratch and to build something proprietary. In contrast, in LwM2M, OMA-DM, and TR-369 USP, a firmware upgrade is one of the basic functionalities of the protocol. It is all an object/instance (depending on the protocol) which is set and pre-defined, including what needs to happen and what needs to be reported during a firmware update. The bottom line is that MQTT is not actually a Device Management protocol, even though some devices and platforms are building proprietary Device Management functionalities out of it. LwM2M and OMA-DM, in contrast, are used for Device Management, while TR-369 USP is the most mature option for Device Management.

- **Response Time:** MQTT takes the lead. Response time for MQTT is very fast – for example, one can increase speakers' volume and stop when the right volume is achieved. With session-based protocols, this is not possible. LwM2M is quite fast as well, and it can take under a second for a command to get to the other side. With OMA-DM and TR-369 USP, it can take as much as 10 seconds for a transaction. Dimming a light with TR-369 USP, for example, might be frustrating. Upon pressing "Dim Down" once and not getting a response, a user may then press it ten more times, which will then result in an overshoot and dim the light far more than the required result. Therefore, for the types of functionalities where a fast response or real-time response is necessary, MQTT is the best option – particularly for sensors and small, unconstrained devices. LwM2M is similar but slightly slower, and it can manage small gateways (or collectors) with a few devices connected to it. OMA-DM is ideal for mobile gateways, for telematics, while TR-369 USP is better suited for fixed and complex gateways.

4.2 Functionality

	MQTT	LwM2M	OMA-DM	TR-069
Push	✓	✓	✓	✓
Discovery / Prov	✗ (Unstructured)	✓	✓	✓
Get	✓	✓	✓	✓
Set	✓	✓	✓	✓
Boot / Reset	✗ (Proprietary)	✓	✓	✓
Diagnostics	✗	✗	✗	✓
Change Notification	✓	✓	✓	✓
App Management	✗	✗	✓	✓
Lock / Wipe	✗	✓	✓	✗
Advanced Security	Basic	✓ (Advanced / Cert.)	✓ (Advanced / Cert.)	✓ (Advanced / Cert.)
Comm. Reliability	Configurable	Configurable	High	High

Figure 4: Standard Protocols Functionality Comparison

The above table illustrates several key points about protocol functionality:

- **"Push":** Because all protocols have the "Push" functionality, all of them enable the server to initiate a session with the device (or rather ask the device to initiate the session upon server demand). All protocols have the option for new devices discovery and provisioning. However, because MQTT is unstructured, the provisioning protocol must be written independently. In the more structured protocols, provisioning is a very basic skill.

- **"Boot and Reset":** For structured protocols this is pre-defined, whereas one needs to do it independently in MQTT.

- **Diagnostics:** TR-369 USP is a highly diagnostic protocol with many sub-protocols contained within it.

- **Change Notification:** All protocols have notification capabilities. However, MQTT requires the server to be more proactive. For example, in a water meter you want to be able to read the usage value periodically, but when leakage is detected (i.e. the meter reads excessive water consumption) the server needs to get real time notification.

- **Application Management:** Because OMA-DM was designed for mobile phones, it is the most capable protocol when it comes to application management. TR-369 USP facilitates some level of application management, while LwM2M and MQTT are simply too basic for this functionality. Protocols that are from the Open Mobile Alliance have “Lock” and “Wipe” as their most basic skill, while TR-369 USP and MQTT are not designed to do that. (With TR-369 USP, however, a work-around is available.)

- **Security:** While all protocols are somewhat secured, some have a greater degree of security. TR-369 USP is highly secured because it was designed for Telcos, as was OMA-DM. LwM2M supports certificates, so one can send certificates and verify who is on the other side. With MQTT, there is the security resulting from the encryption of the transaction, and one can run it over VPN and make it secured. However, using MQTT in the highly secured mode results in loss of speed, lightness and functionality. One can build a very secure protocol over MQTT, but it would be as slow as some of the other protocols. When it comes to the reliability of communication (QoS) and making sure that a message has reached the other end, was understood, and was acted upon, TR-369 USP and OMA-DM are the protocols that were designed for this kind of functionality. LwM2M has three levels of “Use Mode” that one can define.

- **Standardization:** For session-based protocols, there are widely accepted standards, regulated by governing bodies. As a result, every client, or every device that is employing them, can connect basically to any server without too much modification. There are some advanced functionalities and some proprietary functionalities in all of them but the basic set – “Set,” “Get,” “Provision,” “Firmware Update,” “Reboot,” – these basic functionalities are common across the board.

5. Selecting Clients

This chapter briefly examines how to choose the right client among the variety of standard and open clients available in the market.

5.1 Open Source Clients

There is a variety of open source clients available for device manufacturers, and most of these open source clients comply with the regulations. These clients are quite good for educational purposes, experimentation or even small-scale manufacturers. However, for device manufacturers, these open source clients are simply too basic. They comply with the rules, but they are missing some key functionality, are not optimized for transaction and are not fit for mass production.

5.2 Friendly's Carrier-Grade Embedded Clients

Friendly's years of experience in building and improving servers and platforms for Device Management resulted in deep knowledge and understanding of what is expected from a client. When some of Friendly's customers approached the company asking us to provide them with clients, we decided to develop our own carrier-grade clients based on our expertise.

A carrier-grade client has more features than a basic client does. While each of the protocols discussed in this paper has a basic feature set as well as a more advanced (but also regulated) feature set, there are also unregulated feature sets that provide additional functionality – if both the server and client support them. Carrier-grade clients support such advanced features. Friendly leveraged its experience to ensure that the clients are optimized for footprint, memory and transaction time and load. This is significant because when it takes a client more time to respond, it uses more resources and creates a heavier load on the server end. When deploying thousands or even millions of devices on a server, even a fraction of a second per transaction may mean heightened failure probability of the server.

Customers who buy Friendly's clients receive upgrades when the standards change, and new features are added, as well as ongoing service and support. Customers also receive server access for their R&D teams to facilitate deployment.

6. An Example of IoT Architecture Applied to Agriculture

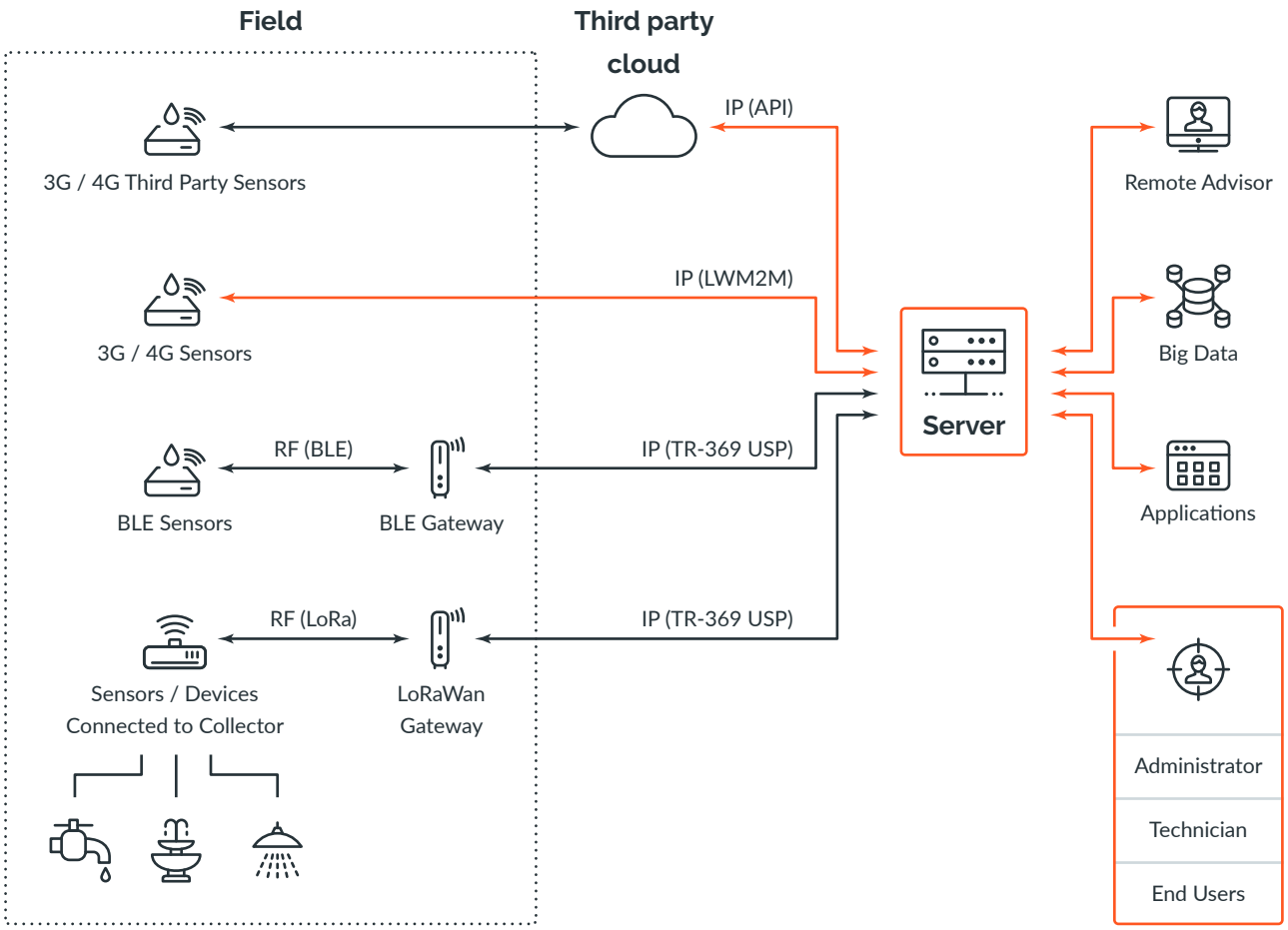

Figure 5: Sample IoT Installation Architecture

Figure 5 above is an example of possible architecture for an IoT installation, and it provides an example of the key topics discussed in this paper.

The illustration shows an IoT platform deployment. Elements in orange are part of the IoT platform itself.

- It is important to realize that many of the devices that need to be integrated already have some sort of IoT connectivity (usually MQTT connectivity to proprietary clouds). In this example, a ground humidity sensor may already be installed in the field, or available from an external source. A good IoT platform would need to include a cloud-to-cloud interface to allow connectivity to external systems and devices. This interface would connect in what is called to the API of the external system (for example through web sockets or REST API) and get the data from the 3rd party sensor. While the IoT platform does not have full Device Management functionality with such devices, one can still "get" ground humidity data or "set" an irrigation on or off.

- Devices that already have a protocol such as LwM2M, can be connected to the platform and be directly provisioned, controlled and configured. These devices have both “Set” and “Get” interface as well as Device Management functionality.

- Many of these installations work through a gateway, which makes the architecture more complex. In the agricultural industry, specifically, where large areas are involved, often devices operate over an RF protocol such as BLE for greenhouses or LoRaWAN for fields.

- A low-cost solution in some cases involves a farmer “driving” through the fields using a mobile gateway (often a tablet or phone) to collect data locally off the various sensors deployed. In this case, OMA-DM protocol would probably be the best choice for managing the gateway, and the sensors would be connected through some sort of RF.

- A common deployment of agriculture IoT involves a collector – a device that allows after-market IoT deployment. A collector is connected to devices directly and supports legacy agricultural devices from the pre-IoT era that might have I^2C, SPI, or other type of wired bus. On the northbound, these collectors often communicate to a gateway over non-IP IoT protocols like such as LoRaWAN. The gateway then is getting managed by the platform over protocols such as TR-369 USP, but may also send the sensor data over other IoT means (LoRa server, for example). Gateway manufacturers are approaching Friendly asking for TR-069 client for their gateways. This is important since the LoRa stack only defines data transactions, and is not concerned (yet) with gateway management.

7. Understanding IoT Platforms

Figure 2 below illustrates a typical example of an IoT platform managing both data and device management. It includes sensors, devices, and gateways. To support third party connectivity, it includes an interface through a smart layer to other third-party clouds. On the server side, one can see the big data, monitoring, and support portal. An administrator portal allows management of the assets, and there is a unified API that allows to control any device with a common API.

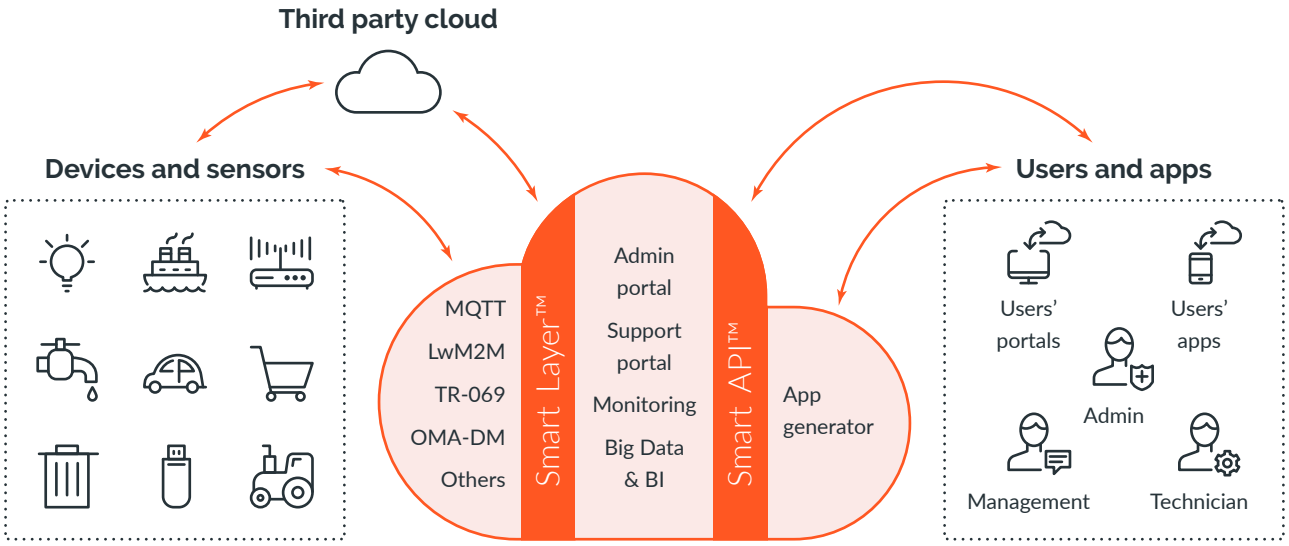

Figure 6: Sample IoT Platform

For deploying IoT as a service, one may want to avoid the need to write a different application for every deployment and device. By using the smart API, one can write an application once, being agnostic of the connectivity to the device deployed. An Application Generator consuming this API can make things even simpler, allowing to write or adjust customizable application quickly and simply without the need for coding.

Figure 7 below shows the Friendly One-IoT™ platform:

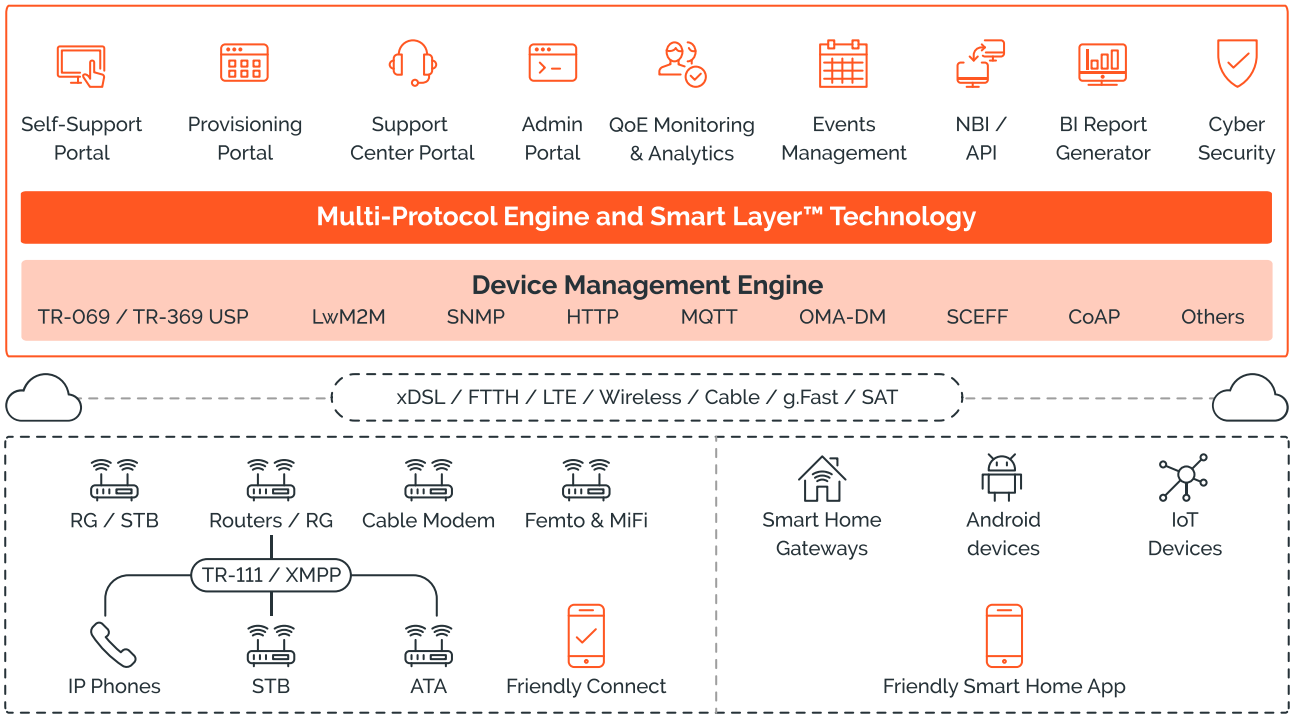

Figure 7: Friendly One-IoT™ Platform Architecture

On the **southbound end**, to the devices, there is an interface through a smart layer where you can have any of the protocols described in this paper or even proprietary protocols

On the **northbound end**, the smart API makes it easy to write a Smart City application for example to control a variety of street light. All that is needed is to write a "Turn on the Lights" command, and the platform would know for each light whether it is LoRa connected light, LwM2M or perhaps a proprietary cloud-to-cloud connected light.

The architecture also includes an **application generator**, which is required for large-scale installations. In a Smart City scenario, for example, there are a variety of gardens, each of which needs its own application. Without an application generator, every garden would require its own specifically written application. With an application builder, however, it is easy to customize a generic application for each garden on the fly, and to customize it later as needed as well. One can do a basic wizard or drag and drop setting and create specific applications for use on a mobile or fixed device.

The **Administrator Console** allows for facilitating management, configuration and provisioning of all of devices through group actions, monitoring and inventory control. It provides visibility of all devices that are on a network (and there may be millions of devices) and management of all of them together, in groups, or each one individually. From the admin console one can define KPIs for monitoring, define events and alarms, generate reports, and manage files.

When managing a specific CPE, the admin can get a full data model of the device and use the collapse and expand options to see every parameter for each device on a network. It is possible to trace a device, reboot it, and reset to factory settings, with the click of a button. When a device uses a standardized protocol, it is easy to run any of these functionalities because, by definition, they are supported. This includes re-provisioning and replacing a device.

For managing groups of devices, the admin can adjust group functionalities. It is also possible to collect big data and reports. One can specify KPIs, for example, by defining, "I want to monitor one of these parameters every hour across all of the devices from this model on my network, or across all devices on a specific street, or for a specific installation." With advanced Device Management, one can specify a group of devices to be updated together based on model, geography, firmware version, or any filter that can be added to define a group. A set of actions can then be selected for the group activity (such as firmware update). For example, one can define that for every device in a group perform the following actions in sequence:

- Reset
- Back up configuration file
- Do firmware update
- Factory-reset
- Restore the configuration file

The script can be scheduled to run on certain days and hours, such as between 1:00 a.m. and 2:00 a.m. every Tuesday. The script can then run on 1,000 devices at one time, only on online devices, and stop and report if more than 5% of the devices fail, or/and stop if a specific action fails. A condition can also be set on when to retry, if at all, etc. One can also follow the progress of a group update, and learn the reasons for failure (where relevant). The advantages of such a platform become clear when you contrast this with a functionality of a basic IoT platform where no Device Management is available in which a firmware update would be to simply send the FTP for the firmware file and ask the device to report to you once it has finished.

Here are some of the other components that the architecture includes:

- **Self-Support Portal:** for basic configuration or problem solving
- **Provisioning Portal:** for adding or registering new devices seamlessly
- **Call Center:** technical support call center for adding the service part to "IoT as a service"

- **QoE:** monitoring and big data analysis for identifying what is the quality of the connectivity to the devices, and obtaining some business intelligence and advanced management, including specifying custom thresholds and alerts for individual devices or groups of devices
- **NBI:** for CRM and other back-office applications and for external applications
- BI Report generator
- **Cyber Security**: real time monitoring of the configuration of devices and proactive detection of cyber-attacks before they happen. The same module can be used to detect other non-security related anomalies in operation and devices. This can be done over the existing device data model or through dedicated embedded agent
- Event Management
- **IoT Builder**: build your own applications using a simple wizard without having to write code. The wizard has various UI and functionalities such as mapping, rules, actions, buttons, monitoring, etc.

The platform also has backward interface compatibility to older protocols such as SNMP, file-based HTTP, and more.

8. IoT as a Service

We foresee that most IoT uses are going to be IoT as a service. Currently many IoT installations are more like M2M, where some automation functionalities are locally defined and monitored. However, more and more applications are based on an IoT infrastructure from service providers (for example deployment of LoRaWAN networks), where by definition, the IoT needs to be serviceable. If you provide a service, you need a Technical support **call center**. Using the network device management capabilities, a call center can look at specific devices and perform tier support for problem solving. For example, change a password, remote reset the device, or find out what are the anomalies or suspicious parameters related to the exhibited problems. A call center can get history of a device, handle firmware upgrades for individual devices, change device configuration and parameters, and it see the entire data model. Whatever is exposed on the device is visible to a customer service representative. There can be different levels and skills of technical support personnel, and one can specify that some of the technicians should not be able to conduct all functionality, but just the basic ones. Experts on the other hand can receive greater access. This is all part of the customer service for a good IoT platform.

Another aspect of device management for IoT as a service can be preventive maintenance. Devices can for example be monitored and if a specific problem occur (such as memory problems) set to automatically reset (or other predefined actions). This can ensure an IoT network is maintained in good health even before issues occur and escalated to the call center.

9. Monitoring Big Data

Big Data or BI and monitoring is a major part of every IoT platform because an IoT platform is about the data. Friendly Technologies, for example, implemented highly professional BI and reporting into its One-IoT™ platform. There are APIs and reports that can be made available for external, professional, proprietary, or existing tools. You can get data for every parameter in the Friendly Technologies system, on every frequency, for up to millions of devices and data points.

10. Summary

IoT and Device Management platforms offer significant benefits. These platforms are important because they do not simply manage the data of your IoT but also manage the devices on your IoT network.

Here are some of the key points to keep in mind:

- Device Management platforms offer unique functionality such as alarms and notifications – not just from sensors and devices, but also from gateways, so you can know when something is or about to go wrong – from the possibility of a cyber-attack, to a device's battery dying.
- It is best to use standardized protocols and servers because experience and knowledge matter - and there is value to maturity.
- Use carrier-class devices, because they work at scale. If you become successful and instead of five hundred installations you reach five thousand, or fifty thousand, you will run into problems if you use something that is not carrier grade.
- Be backward compatible from the beginning. An IoT platform should have admin, and service portals in addition to big data, BI, applications, and APIs to external professional services such as billing, a CRM, or other back office applications.
- For security, cost, and other consideration, it may be preferable to run on-premises or on a private cloud, and not just in a public cloud. Although public cloud is a popular option, when you use a public cloud you share your profits with your cloud infrastructure provider. If you are a service provider, you want platforms that are oriented to service providers – platforms that know your needs and specific requirements.

About Friendly Technologies

Friendly Technologies is a leading provider of carrier-grade platforms for IoT, Smart Home and TR-069 device management. Our advanced solutions enable unified administration of all types of services on one platform.

Friendly has been providing device management solutions to carriers and service providers since 2007. When IoT and the Smart Home emerged, Friendly Technologies launched its IoT platform and full solution for the Smart Home. Leveraging our expertise in the device management field, Friendly extended its offering to IoT and the Smart Home markets.

Friendly's products support LwM2M, MQTT, OMA-DM, TR-069/TR-369 USP, CoAP, and HTTP protocols, as well as proprietary protocols. We provide solutions via NB-IoT, 5G, LoRA, FTTH, and DSL networks.

Friendly's device management platform allows our customers to connect and provision new devices automatically, monitor KPIs, configure and update firmware remotely, collect and analyze data, and streamline the support they provide for subscribers.

Hundreds of providers worldwide trust Friendly's device management and IoT solutions. Our products are scalable and future-ready, enabling our customers to reduce both CAPEX and OPEX and generate new revenue streams.

Among Our Customers

Don’t hesitate to contact us for a further information.

International Office
Friendly Technologies Ltd.
Tel: +972-3-753-9000
Email: sales.hq@friendly-tech.com

European Office
Friendly Technologies Ukraine
Email: ukraine@friendly-tech.com

Latin America Office
Friendly Technologies Colombia
Email: colombia@friendly-tech.com

US Office
Friendly Technologies, Inc.
Tel: +866-447-8640
From Canada: +888-805-7691
Email: friendly.usa@friendly-tech.com

China Office
Friendly Technologies, China
Tel:+86-755-61931701
Email: yuanjie@friendly-tech.cn