Coordinated autonomic loops for target identification, load and error-aware Device Management for the IoT

Neil Ayeb, Eric Rutten, Sebastien Bolle, Thierry Coupaye, Marc Douet

Coordinated autonomic loops for target identification, load and error-aware Device Management for the IoT

Neil Ayeb*, Eric Rutten†, Sebastien Bolle*, Thierry Coupaye* and Marc Douet*

*Orange Labs, Meylan, France
*Email: firstname.lastname@orange.com

†Univ. Grenoble Alpes, Inria, CNRS, LIG, F-38000 Grenoble France
†Email: firstname.lastname@inria.fr

***Abstract*—With the expansion of Internet of Things (IoT) that relies on heterogeneous, dynamic, and massively deployed devices, device management (DM) (i.e., remote administration such as firmware update, configuration, troubleshooting and tracking) is required for proper quality of service and user experience, deployment of new functions, bug corrections and security patches distribution.**

Existing industrial DM platforms and approaches do not suit IoT devices and are already showing their limits with a few static home devices (e.g., routers, TV Decoders). Indeed, undetected buggy firmware deployment and manual target device identification are common issues in existing systems. Besides, these platforms are manually operated by experts (e.g., system administrators) and require extensive knowledge and skills. Such approaches cannot be applied on massive and diverse devices forming the IoT.

To tackle these issues, our work in an industrial research context proposes to apply autonomic computing to DM platforms operation and impact tracking. Specifically, our contribution relies on of automated device targeting (i.e., aiming only suitable devices) and impact-aware DM (i.e., error and anomalies detection preceding patch generalization on all suitable devices of a given fleet). Our solution is composed of three coordinated autonomic loops and allows more accurate and faster irregularity diagnosis, vertical scaling along with simpler IoT DM platform administration.

For experimental validation, we developed a prototype that demonstrates encouraging results compared to simulated legacy telecommunication operator approaches (namely Orange).

Keywords–device management, multiple loop cooperation, internet of things, firmware update, configuration management.

I. Context & Motivation

A. Device Management

Device Management (DM) consists of remote (and potentially massive) operations on a fleet of deployed devices. Managed devices include, but are not limited to, workstations, broadband or IoT gateways and smartphones. After the wide usage of Blackberry and smartphones later-on in business context, enterprises used Mobile Device Management (MDM) for application remote installing, configuration provisioning and over-the-air (OTA) software updates [1].

Historically, DM solutions targeted workstation for configuration management, application provisioning and OS patching. Afterwards, with high speed (e.g., Digital Subscriber Line, Cable) internet accesses widespread, routers and modems (a.k.a Boxes) required user specific configurations and regular firmware updates to enhance Consumer Premises Equipment (CPE) lifespan. This led to the creation of an industry consortium, the Broadband Forum [2], (initially ADSL Forum) that aim to standardize network technologies and protocols for internet access over phone lines, then progressively integrated *home device management* specifications and architectures among other topics.

With the rise in complexity of services and device computing power, later Telco DM solutions started incorporating remote troubleshooting features.

B. DM Features

We define DM as initial and 'in-life', remote, firmware updates (Maintenance) configuration of devices (i.e., Provisioning), probe data collection (Monitoring), and troubleshooting (Assistance) [3]. Maintenance lets system administrators push firmwares on the entirety or specific parts of a given device fleet of a given device fleet (e.g., patching a security issue with a new firmware, deploying a new feature for beta test members). It was the core feature for early solutions. Historically, DM was mainly about remote firmware updates. With the increasing complexity of devices and maintenance costs, solutions started incorporating new features such as those we mentioned above. Provisioning incorporates new services (de)activation, and equipment behavior modification by editing a given device data-model (e.g., change its data platform URL, enable new sensors of a modular peripheral, pair with a nearby device). Monitoring grants log and runtime data to be pushed using a DM protocol to a central platform for data analysis (e.g., using QoS measures to detect network congestion or average resource consumption per device category). Assistance empowers system administrator to remotely execute diagnostic commands on devices aiming for troubleshooting without physical intervention (e.g., triggering reboots, factory configuration resets aiming to fix a peripheral).

DM is crucial a continuous, correct functioning of devices, while ensuring proper Quality of Service (QoS) for end users or business partners. Yet faulty DM operations can cause consequent losses for companies, especially when they target a whole fleet of devices. In october 2019, Google pushed a faulty firmware to a part of its home voice assistants *Google Home* and *Google Mini* fleet. After a reboot, devices became unable to boot or were locked in an infinite loop. This incident had a negative economic impact on Google since all devices were replaced free of charge [4]. If we suppose that 1% of the fleet (i.e., 52 Millions at the end of Q4 2018) [5] was affected, it would represent 520,000 devices bricked to be potentially replaced, plus shipping costs. Moreover, HP-Enterprise (HPE)

and Dell EMC deployed a critical firmware update designed by their enterprise SSD manufacturer Western Digital. Indeed, a faulty firmware was integrated within these drives at release time (2015). Without fix, data loss become basically inevitable after 32768 hours of running [6] [7]. Beside fault mitigation, DM can be used for performance enhancement and new features deployment. Indeed, *Tesla* electric vehicles received in November 2019 an OTA (i.e., Over-the-Air meaning via 4G-LTE Networks) software update that enhanced the peak output power of *Model S* engine by 37kW. Besides, Tesla's firmwares constantly improve self-driving and charging capabilities. Telecommunication operators use some DM features of their home and enterprise internet gateways (a.k.a. boxes) for remote troubleshooting when customers call after-sales service for technical assistance. It allows pushing new firmwares and configurations to its deployed fleets of home devices (e.g., Broadband and Fiber Gateways, TV decoders, Wi-Fi Extenders). These experiences show how crucial DM is for businesses and give a peak about its evolution with Internet of Things widespread.

IoT relies on massive deployment of various devices (e.g., sensors, gateways, actuators..). However, they are usually heterogeneous regarding their computing, storage, and communication capabilities (i.e., simple sensors configured to push data vs. peripherals with high computing power) and environments (i.e., mobile on-battery devices vs. rather powerful fixed gateways). IoT Services can rely on multiple devices from distinct manufacturers and owners. This entails collaborative DM Platforms for service provisioning, troubleshooting and configuration.

C. Challenges and contributions

These characteristics, compared to legacy DM, involve several challenges regarding IoT Device Management.

- Heterogeneity: From few types of devices to numerous ones and various environments (connectivity, nearby devices), computing, storage and networking capabilities. IoT allows services to use multiple devices in contrast with single owner traditional objects.
- Dynamicity: From internal states (battery, computing load, running services, network conditions) that are rather stable to versatile ones that changes over time.
- Interoperability: From isolated technical solutions (i.e., one DM platform per device type) to multi-platform devices enabling multiple interdependent services.
- Scalability: From a few 'things' managed by centralized designs to massive amounts of devices.

In this study, our goal is DM platform operation automation, therefore tackling device **Heterogeneity** and environment **Dynamicity** by adjusting execution speed to capabilities of devices and infrastructure (i.e., hardware, current load and network congestion). Our adaptation strategy relies on operational metrics such as monitored DM operation execution error count or infrastructure response time. Furthermore, a step towards **Scalability** management is introduced in this paper via vertical scaling [8].

Operating DM solutions require consequent efforts and expertise. For instance, according to interviews realized with DM teams, updating a fleet of residential routers at Orange is done as follows:

First, the DM system should be given (by its administrator) a firmware and its target (i.e., subset of devices in this version of this manufacturer) that will be processed. For each DM operation, it is potentially required to add specific parameters (e.g., higher retry tolerance, variable firmwares depending on subscribed services or owners). Once launched, firmware installation is remotely done for each device. During that phase, active monitoring is performed remotely by administrators for fault detection and tracking. After complete target migration to the new firmware, a rollback can potentially be triggered if unexpected device behavior is reported by users. This typical workflow cannot be adequate for IoT DM if the previously mentioned challenges are taken into consideration due to high complexity and failure risk.

In order to address the above-mentioned IoT DM challenges, this paper makes the following architectural and experimental contributions:

- A **Coordinated multi-loop autonomic architecture** for IoT DM;
- An **Operation Generation & Target Identification** loop for automatic target (i.e., subset of devices to process) identification and operation launching;
- A **Decomposition, Enforcement and Tracking** loop for DM operations execution and monitoring;
- A **Speed Regulation** loop for batch size variation (i.e., amount of devices processed in each **Decomposition, Enforcement and Tracking** loop iteration) depending on anomalies and infrastructure load;
- A **Proof of Concept** for speed regulation and impact assessment validation.

The rest of this paper is organized as follows. Section 2 describes existing research and industrial work. Section 3 introduces our contribution regarding Autonomic IoT DM. Section 4 discusses our proof-of-concept and experimental results. Finally, Section 5 presents our conclusion and perspectives.

II. Related Work

We propose to analyze the different categories of DM solutions, i.e. Home, Mobile, Workstation and IoT, to compare their proposed features as well as technical implementations.

Our survey on existing work led us to conclude that the main objective of authors is to optimize executing firmware updates on constrained embedded boards, therefore focusing on the process itself and not fleet management or DM platform operation. These devices are often working using battery power and do not offer much computing power. Therefore, update process adaptation (e.g., optimization [9], [10], [11], securing [12],[13],[14]) is explored for such devices. Existing standard DM protocol are also studied and compared for IoT usecases [15],[16],[17].

To the best of our knowledge, no existing research work aims to automate DM platforms operation for IoT management.

	Orange LiveObjects	Amazon Web Services IoT	Microsoft Azure Iot Hub	Bosch IoT	IBM Watson
Firmware Update	✓	✓	✓	✓	✓
Configuration Update	✓	✓	✓	✓	✓
Standard DM Protocol	LWM2M + Custom	Custom	Custom	CWMP - OMA-DM - LWM2M	Custom
Campaigns	✓	✓	✓	✓	✓
Execution Speed Regulation	✗	✗	✗	✗	✗
Dynamic Target	Partial	✓	✓	Partial	✗
Reactivity to DM ops. errors	✗	Progress Reporting	Progress Reporting	Progress Reporting	Manual

TABLE I. IoT Platform DM Capabilities Survey

A. Industrial Solutions Analysis

From our analysis of existing Home, Mobile, Workstation, and IoT DM solutions and their features we observe that:

- Home (e.g., Internet residential gateways and TV decoders) management does not offer application stores except Android TV devices therefore partially resembling IoT management in that point. The main difference with the IoT is that home devices are always plugged in AC power, usually connected with reliable network connections and embed some auto-diagnosis features.
- MDM (Mobile Device Management) differs from IoT DM by a significant firmware fragmentation [18] and user-triggered updates. Android will also allow seamless transition (i.e., no service interruption) from one firmware to another [19] in its next release. Such advanced mechanisms cannot easily be implemented in constrained IoT devices. For instance, IoT hardware limitations does not allow the separation between applications (Google Store) and kernel-OS packages (Android ROM).
- Workstation DM, (according to our internal survey at Orange), allows operating system (OS) updates and configurations installation while machines are used with little acceptable performance degradation. These updates are applied outside work hours by a remote reboot command (or by waiting for end users to reboot). IoT updates could be theoretically differed but firmware update implementations usually triggers a reboot at the end. Even though some IoT devices run UNIX kernels or micro-kernels, they do not include package management like their desktop and server counterparts.

Configuration updates and remote troubleshooting of IoT devices has some similarities but also significant differences with home, mobile and workstation management. The key difference between those is firmware nature (OS + applications for some, OS only for others) and update mechanisms (instant reboot, firmware developed by manufacturer or a third party). Besides, Home and mobile device manufacturers keep their firmware closed source and rarely implement DM API. IoT however is going towards openness via unified standards (e.g., OneM2M) and protocols (e.g., Open Mobile Aliance Lightweight Machine to Machine LWM2M) Due to the aforementioned differences, efforts have been focused to analyze and identify limitations in telecommunication device management, and also investigate Orange current management strategy for its device fleets, in addition to surveying industrial IoT platforms capabilities and features.

1) Telecommunication Operator Home Device Management:

a) Features: Orange current Home DM solution is internally developed. Its based on CWMP [20] protocol (i.e., CPE WAN Management Protocol), also known by the name of its technical specification document: TR-069. It enables remote firmware updates, tracking, troubleshooting, and configuration of Livebox Routers (i.e., Orange's Internet Gateways) and Set-Top-Boxes (TV Decoders and Multimedia Gateways). This solution is currently centralized and manages around 20 Millions of CWMP Devices.

CWMP proprietary platforms (Arris, SagemCom, Axiros) and open-source solutions (e.g., GenieACS [21], FreeACS [22]) exist and are being operated for router and other TR-069 compliant devices management. They cover firmware update and device configuration but do not usually offer advanced mechanisms such as dynamic device groups, operation tracking and history.

b) Operation Analysis: Our study of Orange's strategy for HomeLAN device management shows this behavior: A firmware will be installed on a few devices and these will be manually observed by experts (i.e., who have the ability to interpret probe data, and trace errors). Afterwards, several thousands of devices will be migrated and will stay under observation during approximately ten days among other indicators such as hotline calls for malfunctioning devices. In case of very high amount of reports by users and field-technicians, firmware will be declared non-viable by system administrator, the global operation canceled and processed equipment returned to its original firmware. Otherwise, firmware is set to be installed in the entire fleet.

2) IoT Device Management:

a) Features: In Table I, we compare existing technical IoT platform DM capabilities. Our criteria include minimal DM features (Firmware and Configuration Upgrade), Campaign Launching (i.e., operations on multiple devices of the fleet), error reactions (ability to react to execution abnormal behavior from a platform or devices), speed regulation (how much devices are processed in a given time slot) or dynamic target (entities that includes devices depending on their current hardware or software states).

All of existing platforms incorporate at least firmware and configuration update capabilities. Depending on the platform standard DM protocols can be supported or custom ones can be used. Orange Business Services (OBS) Internet of Things platform Live Objects [23] incorporates DM capabilities. It is targeting rather constrained devices and enables remote firmware update. AWS IoT as Microsoft Azure IoT Hub and Bosch IoT Suite adds to these features dynamic group capabilities, progression tracking of operations and additional

multiple standard DM protocols support for Bosch IoT Suite. However, IBM Watson IoT only support firmware and configuration updates on multiple devices via custom proprietary DM protocols.

b) Operation Analysis: Strategies are defined by system administrators and are specific to usecases. It depends on which devices are managed, what hardware capabilities they have, their network environments and finally how much device anomaly/error/loss is tolerable. To the best of our knowledge, no public documentation regarding such strategies exists.

B. Discussion of existing work

A part of existing research work [9], [10], [11], [12], [13], [14], [15], [16], [17], aims to tackle embedded and security constraints regarding firmware update process therefore optimizing DM at a device level. None focus on large fleet management (e.g., numerous IoT gateways, sensors or actuators) and DM platform operation optimization as a whole (e.g., configuring firmware-device association, firmware deployment strategy).

Feature-wise, the previously mentioned approaches and solutions for Home Device Management are not suitable for IoT objects. Indeed, the former are usually very specific designs, thus very efficient for single device types, but unsuitable for heterogeneous device fleets in various and dynamic environments (i.e., constantly changing battery states, computing load, network signal, activated services). In contrast, traditional home network devices are usually plugged-in AC power, mostly wired to a wide-area-network access, and managed by a single specific owner/vendor DM solution. Industrial solutions however such as Amazon IoT [24], Azure IoT Hub [25], Bosch IoT Suite [26], and Orange LiveObjects [23] offer many characteristics that suit IoT DM such as light communication cost implementation, multi-protocol and various device type compatibility.

However, performing large operations still require the intervention of experts. Operation and configuration is done manually by them (as with Home DM). Indeed, these platforms do not implement mechanisms that automate target identification in a fleet, or allow error detection and mitigation operation execution and tracking. Moreover, device states (e.g., load, network, software) are not taken into consideration for management operations and fleet firmware update. The dynamicity of device states implies several specific configurations to be deployed on subsets of a given device fleet, even in case of homogeneous set of devices. Thus, active and complex reconfiguration of such DM solutions is required for correct and optimal functioning.

These characteristics lead us towards investigating autonomic computing approaches for IoT DM platform operation and configuration while also automating anomaly detection and mitigation. We aim to add an autonomic management layer on top of existing DM platform to enhance their capabilities and automate their operation.

III. Autonomic IoT DM Managers

We propose an architecture based on three cooperating autonomic loops respectively for 'operation launching and target identification', 'DM operations speed regulation', and 'on-device execution while detecting anomalies'. Our proposal is platform and protocol agnostic, therefore masking existing DM platforms complexity and specificity. This allows integration of diverse objects while extending and interfacing with various DM solutions.

A. Overview

IoT device firmware get released by manufacturers during device's commercialization and support period. Configurations are developed and pushed when required (e.g., new service, security flaw patching etc..) by the system administrator. These are not meant to be systematically applied to every compatible device except for security updates that should be generalized as soon as possible.

Based on device and firmware information (extracted from their datamodels), it is possible to infer whether a device should be updated or not. For instance, when on low battery or poor network conditions, firmware updates should be avoided to prevent a corrupted installation that would render the device definitely out-of-order. Another example is only targeting relevant devices for a minor firmware or configuration update (i.e., fixes a bug when certain features are enabled or used). The main goal is avoiding service interruption and serious dysfunctions risks induced by DM operations when they are not critical. Besides, if faulty DM operations (e.g., bugged firmwares or configurations) are performed on a part of the fleet or all of it. Consequences can be (but are not limited to) lower QoS, abnormal behavior and device rendered unable to function again without manual flashing or replacement. To avoid generalization of such firmwares and configurations, we propose active monitoring of DM Operations impact.

To sum up, we aim to integrate part of a system administrator expertise in an autonomic management layer that will enhance DM platform operation. Thus, we identify two key features for a smarter IoT DM:

- Automatic operation launching and target (devices) identification;
- Automatic anomaly tracking on both of devices and infrastructure (i.e., hardware and network hosting the DM platform).

These features are defined to take on three of the main challenges induced by the massive deployment of IoT devices. Indeed, our architecture tackles the **Heterogeneity** of IoT by allowing multiple device types to be handled by the same autonomic managers by abstracting the concept DM Command (i.e., protocol specific elementary) and allow **heterogeneous** device types to be handled by the same autonomic managers. Besides, **Dynamicity** of IoT devices conditions is also addressed by our proposal in two ways: using device state active checking for target identification and operation launching (e.g., computing power at a given time, running services, paired devices, network signal, movement, interference), and anomaly tracking also allow our proposal to tackle **Dynamicity** via active monitoring of defined QoS measures during progressive operation execution on the fleet. Indeed, deployment speed is regulated according to operation metrics (e.g., errors, warning). **Scalability** is managed via our approach thanks to its ability to vertically scale (i.e., use more available computing power to increase execution speed [8]) depending on infrastructure load.

B. Multi-Loop IoT Device Management Architecture

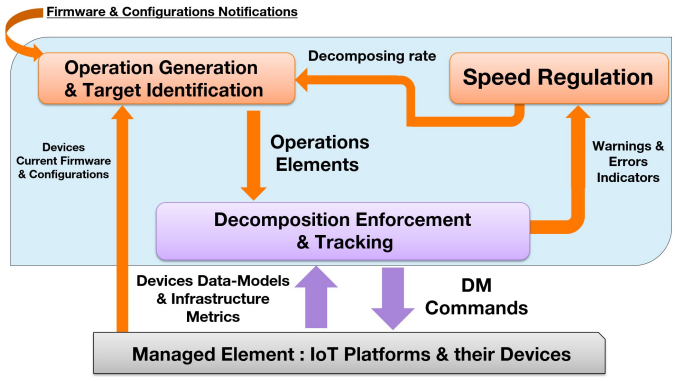

Figure 1. Global multi-loop architecture for DM

We propose an autonomic control system that will automatically operate some features of DM platforms, namely operation generation, target identification, anomaly detection (e.g., execution errors, infrastructure overload), processing speed variation. It is based on multiple coordinated control loops that react to execution errors and warnings faster than existing human-based approaches.

The architecture is able to tackle IoT challenges (i.e., numerous devices with different context, states and capabilities). Indeed, it automatically triggers DM operations and identifies suitable devices to process (i.e., correct initial firmware, battery and network status). Moreover, it takes into consideration hardware load and network congestion.

The system aims towards one global goal: 'Keeping a device fleet compliant and up to date'. It is composed of three autonomic loops (see Figure 1). First, *Operation Generation & Target Identification* gets devices datamodels via IoT Platforms. This allows automatic target identification and operation launching. Second, speed execution variation is operated by the *Speed Regulation* loop that decides, depending on progression, phase, errors count, and infrastructure response time, among other measures, which speed should be forwarded to the Operation Generation & Target Identification manager. Finally, the *Decomposition Enforcement and Tracking* manager actually sends commands to devices and collects execution data and logs that are compiled and sent to the Speed Regulation loop to compute.

We distinguish two levels of managers. High level ones (colored in orange in Figure 1) that are centralized and concentrate all data for decision-making (operation generation & target identification, speed regulation). Low level managers (colored in purple in Figure 1) are meant to be instantiated multiple times for horizontal scaling.

1) Managed Element: DM platforms and their managed device fleets, each containing one (or more) device type (e.g., Netatmo Home Weather Stations, Philips Hue Devices) represent the managed element of our proposed autonomic control system.

2) Monitoring: The autonomic management system observes these data:

- Firmware & Configuration Notifications: they contain information (e.g., Type or Criticality, Installation requirements) used for prioritization and target identification;
- Device Hardware and Software states (extracted from their datamodel instances), for error and warning detection;
- Infrastructure Metrics (e.g., DM Servers API's response times) for overloading avoidance;
- Optional: External business information such as amount of hotline calls.

Firmware information is provided by developers. It takes the form of a description file manifest that contains information such as Type (Critical, Major, Minor, Hotfix) or installation requirements (e.g., migration path: v1.1 to v1.2 to v2.0, minimum battery level). This information is used by our autonomic system to target the right subset of devices that should be receiving the DM Operation.

In order to accurately detect QoS variation for warning diagnosis and error mitigation, we defined a collection of commonly available metrics in devices datamodels. These include an average CPU usage, RAM load, and network interface occupation. In addition, for accuracy's sake, we propose a set of device-type related measures. For instance, an IP Camera should show a stable video bitrate (within margins).

Infrastructure congestion mitigation is done via response-time observation. In fact, an increase in that measure implies a size reduction of sent DM operations for execution.

3) Effectors: In order to keep a fleet up to date and well-functioning, devices that were targeted will receive a set of DM commands (i.e., elements composing a DM operation) generated by the autonomic IoT DM system.

In the following subsections, each autonomic manager has its input, output, pace and workflow detailed.

C. Decomposition, Enforcement and Tracking

This autonomic loop aims to push DM commands to IoT platforms while enforcing execution policies (e.g., asserting max parallel operations possible on servers, operation prioritization, retry approaches). It is also responsible for tracking data (e.g., logs, server response-times, probe data) aggregation.

a) Input: Three entries are necessary for this loop to run:

- The first input required for this autonomic loop to operate is a set of 'DM Operation Elements'. They target a part of identified devices (e.g., 20% of eligible devices for a firmware installation). These are computed by Operation Generation loop.
- Ongoing devices datamodels are the second input of this loop, allowing device state assessment during execution (e.g., detect whether a *DM Command* is properly executed or not). These states include current firmware version, QoS measures, hardware state (e.g., battery power, CPU load, free memory).
- The last entry is related to infrastructure metrics (e.g., response time, overload alerts, amount of network or platform errors).

We propose two sets of QoS indicators related to managed devices: commonly available probe measures in different standard protocols datamodels (LWM2M [27], USP [28], CWMP [20]) and a set optional of device-type related indicators.

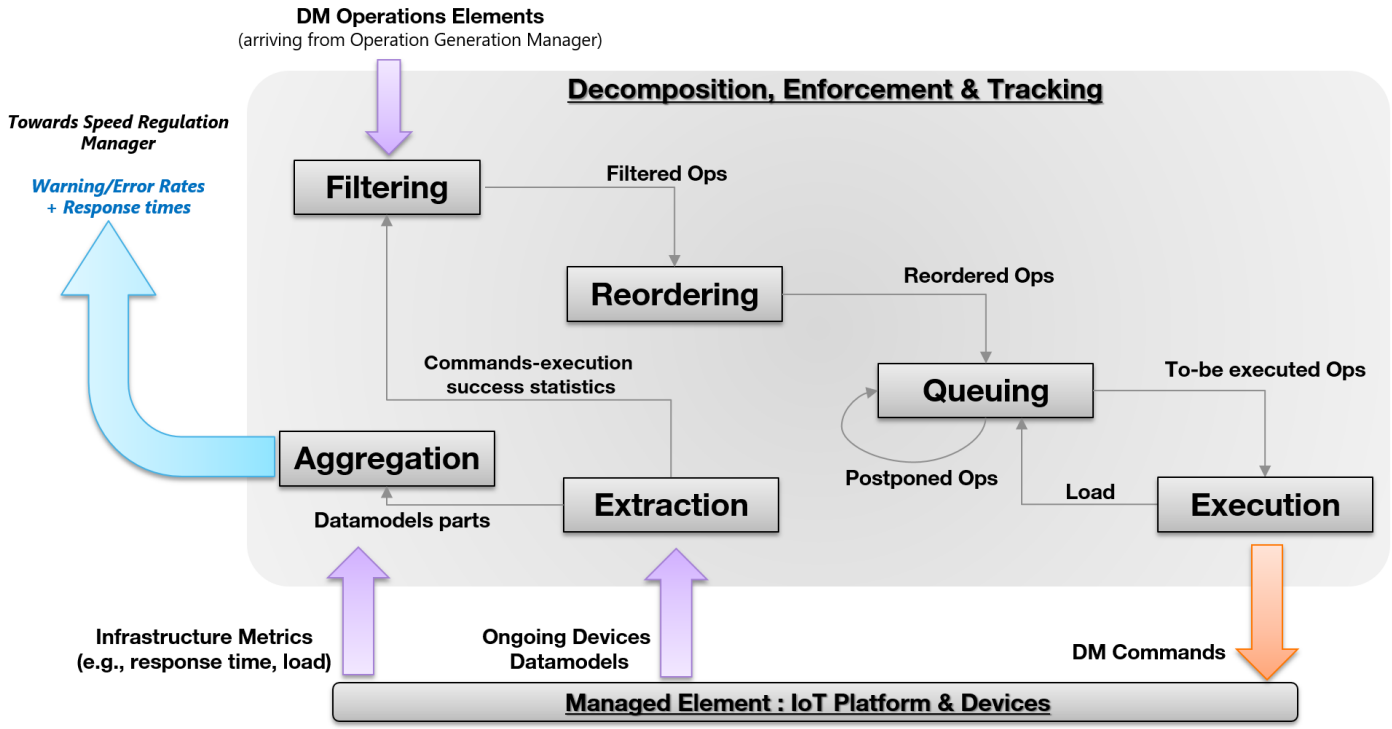

Figure 2. *Decomposition, Enforcement & Tracking* autonomic workflow

First ones are as follows:

- Average CPU usage per time slot (e.g., 6 hours);
- Average RAM usage per time slot;
- Storage utilization;
- Amount of network packets sent, received, errors;
- Network signal strength.

Optional data include (but are not limited to):

- Application QoS measures (e.g., video bitrate, abnormal sensor data);
- CPU usage variation per time slot (e.g., for spike detection).

We aim by analyzing these data for more accurate processing of hard to detect losses of QoS (e.g., slight variations but randomly happening, big spikes on a certain type of environment). These are usually assessed via costly physical interventions from technical services following a customer care call. These trips should only be triggered for mandatory interventions (e.g., battery replacement, hardware failure).

b) Output: Two outputs emanate from this autonomic loop.

- Error and Warning percentages and rates (in blue in the scheme): they indicate if a DM Operation have a negative impact on devices, or if a firmware doesn't install properly on a part of the target. These measures are extracted from devices datamodels and aggregated before being sent to *Speed Regulation* loop. It also integrates QoS measures regarding DM Servers (e.g., infrastructure response-time)
- Protocol specific DM Commands (in orange in the scheme) inferred from ongoing DM Operations Elements. They are sent to devices via one (or more) DM Server APIs.

c) Pace: This autonomic loop keeps running while all awaiting and ongoing DM Operations are not completed or failed.

d) Workflow: Figure 2 provides a global picture of how are the DM Operations Elements processed. Its workflow is composed of the following steps:

- *Filtering*: First module filters out *DM Operations* that does not comply with retry policy. This action avoids loop's overflow due to several unsuccessful operations being queued. This module assesses if on-going *DM Operations* are successful based on *Execution Success Statistics*. For instance, an operation can expire if a certain percentage of its commands fail after several retries. These percentages are extracted from devices datamodel. Operation nature should be taken in consideration. Indeed, a network configuration patch on a fleet may be considered failed if not applied on 100% of the fleet.
- *Reordering*: Afterwards, reordering component proceeds in rearranging operations to be treated regarding their priority. It is either set by the DM Operator for a manual *Operation*, or inferred using loop's knowledge base. We identified the following order:
  1) Critical (e.g., security patch, urgent fix);
  2) Major (e.g., new feature release);
  3) Minor (e.g., non critical bugfix);
  4) Hotfix (e.g., bugfix for rare certain devices in specific environments).

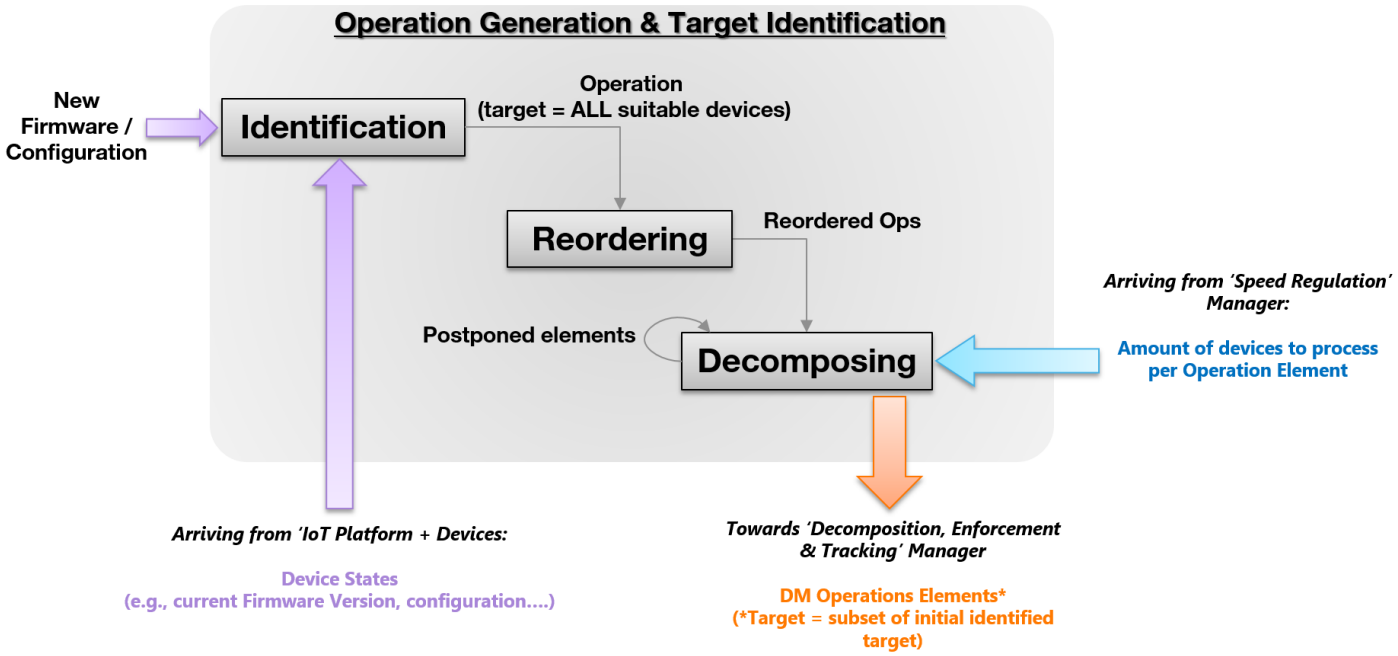

Figure 3. *Operation Generation & Target Identification* autonomic workflow

This procedure is needed in order to avoid higher priority operations to be systematically executed after on-going massive low priority ones (e.g., a security patch on a small set of devices must be pushed before applying a minor hotfix to a complete fleet of sensors).

- *Queuing*: Next module concerns operations Queuing. It sets some operation for later execution depending on node's hardware capabilities. It computes how much parallel operations can be executed and tracked. This mechanism avoids hardware overloading.
- *Execution*: Once to-be-executed operations are identified, they are translated to protocol specific *DM Commands* that are sent by a DM Server to the targeted devices (i.e., Managed Element). For a given autonomic loop, Device Type and DM Protocol are identified (e.g., Indoor Geolocalization Station, LWM2M).
- *Aggregation*: In order to track proper execution of a given *DM Command*, the loop pulls execution-data from devices then Aggregates warning and error percentages. This data is used for two purposes. First, retry policy relies on it to discontinue operations when considered failed. Second, compiled execution data is pushed to *Speed Regulation* Autonomic loop allowing it to regulate DM Operations progression. Errors represent devices that do not respond after updates, while warnings incorporate DM Server response time variation and abnormal device behavior (e.g., wrong values, high memory usage, frequent registration rate).
- *Extraction*: This entity consists of querying devices datamodels and extracting required data from it, current firmware version and impacted features QoS measures for a firmware update operation. For a given device, a datamodel defines its state, hardware capabilities, and software environment.

D. Operation Generation & Target Identification

This autonomic loop is responsible for DM Operation generation (based on the managed fleet state) and decomposition in *DM Operation Elements* that will be forwarded to the *Enforcement, Decomposition & Tracking* loop. Indeed, it is in charge of target identification (i.e., defining the currently suitable devices that will (or need to) receive a given firmware or configuration) while applying computed ongoing processing speed (via decomposed operations size) based on the decision of the Speed Regulation loop.

a) Input: Two events can trigger its activation.

- New firmware or configuration notifications and their description file. This information allows target identification and priority inference.
- Manually sent DM operations (e.g., Update Compliant Weather stations to Firmware 3.1, Enable motion detection on all Proximity Sensors). These are basically notifications in which target is manually defined by DM Operator (or Administrator).

Besides, another input is required for this autonomic loop to fulfill its role. Indeed, the computed amount of devices by *Speed Regulation* loop is injected for operation speeding up or down.

b) Output: There is a single output for this autonomic loop. It consists of DM Operation that aim suitable devices from the fleet (e.g., Devices with correct network conditions).

c) Pace: This autonomic loop keeps monitoring the deivce fleet therefore detecting new devices that are not compliant (new or out-of-date device). If new DM operation is to be executed or ongoing, it keeps passing through all its phases until done.

d) Workflow: Figure 3 details this manager's workflow. It is composed of the following steps:

- *Identification*: Once an input received by the loop, this module triggers target identification based on included information in firmware description. For instance, a given system update may only be applied to devices in the right current version. Another example is minimum battery requirement for patching.
- *Reordering*: Next module is in charge of business SLA application through reordering. Indeed, in an open DM platform (e.g., DM as a Service), contracts with 3rd parties may induce variable SLA agreements. It is different from Decomposition, Enforcement & Tracking Loop reordering module. This one aims to do SLA-based high level prioritization of pending operations, while the lower entity reorders parts of ongoing ones.
- *Decomposing*: Last task treats how fast a *DM Operation* progresses depending on warning and errors by increasing or decreasing its batch size (i.e. amount of device processed each iteration). This amount is based on *Speed Regulation* loop that computes how much devices should to be treated according to execution anomalies rates and IoT platform infrastructure load.

E. Speed Regulation

This loop aims to decide whether an ongoing DM Operation should be accelerated, slowed, halted, or stopped. It takes its decision based on warnings, error rates on devices received by *Decomposition, Enforcement, & Tracking* loop, while also taking into account infrastructure load metrics sent by the managed IoT Platform.

DM Operations are characterized by their 'State' and 'Phase'.

- State: Pending (Not Started), Ongoing (being processed), Aborted (Stopped due to high error rates), Finished (successfully executed)
- Phase: We define three possibilities: Test (beginning of an operation). Cautious (next step, with more devices yet moderate speed). Generalization (fast phase where the goal is to process as many devices as possible). Phases are determined by target percentage that have been processed.

We propose a rather simple algorithm making that choice based on the current phase of an operation. Depending on in which phase it is, the changes of speed will be more or less significant. Thus, error and warning tolerance are lower in *Test* phase. This allows more accurate decision-making for speed regulation: the earlier anomalies are detected, the more drastic is the regulation.

a) Input: Input here is error and warning rates arriving from *Decomposition, Enforcement and Tracking* autonomic loop and infrastructure metrics. Both are used for decomposition rate computation.

b) Output: This loop outputs the right number of suitable devices that should be processed in *Operation Generation & Target Identification* loop based on error and warning rates.

c) Pace: Each execution cycle of this loop is triggered by an operation batch finishing in Decomposition Enforcement & Tracking loop.

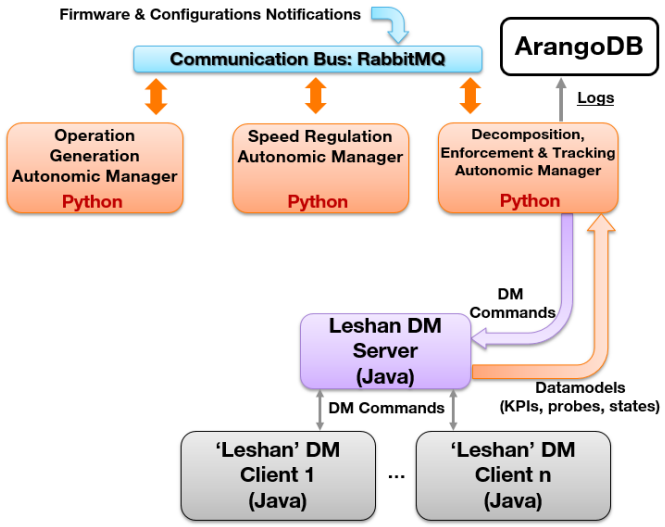

Figure 4. Technical Component Architecture

d) Algorithm: This algorithm works as follows for each operation:

1) If pending: Start Operation
2) If errors > tolerated error rates, Abort
3) Update Phase (depending on progression)
4) Regulate: According to metrics variation and phase : Speedup or Slowdown

Multiple regulation strategies are possible. Depending on risk tolerance, variation can be Linear, Power, Polynomial, Exponential.

IV. Proof Of Concept & Experimental Validation

In this section we detail our experimental setup in order to evaluate our approach capability to regulate speed automatically. First, we provide details regarding the technical architecture of our setup. Afterwards, we present our environment before describing autonomic loop implementation.

A. Implementation & Experimental Results

For this experimental setup, we used Eclipse Foundation open-source DM Client and Server (i.e., Leshan Project) [29]. Each autonomic loop is implemented in the form of a Python script. Inter-loop communication is done via a messaging queue: RabbitMQ. Figure 4 details components interactions. Currently, all software modules run on a single physical server therefore no network latency is present during inter-process communication. During execution, logs are being pushed to a database for further analysis and interpretation. ArangoDB serves as a database for our experimentation, it hosts logs and past execution traces.

Our internal survey of Orange DM approach leads us to model it in three phases as follows. Initial phase aiming few devices (less than 0.01%) of the fleet lasting 48 hours of manual metrics observation. Second phase targets few percents of our fleet (i.e., 3%). It lasts 10 days of execution and surveillance time. Last step is a generalization phase that installs the update in all available devices at a rate of 6.25% of the fleet per 24 hour slot (rate defined by experts as a

	Existing Orange Home DM	Industrial IoT Platforms	Autonomic Enhanced Platform
Operation Launching	Manual	Manual	Automatic
Target Identification	Manual	Manual	Automatic
Protocol Support	Single	Multiple	Multiple (Platform Dependent)
Execution Speed	Static	Static	Dynamic
Anomaly Awareness	✗	Partial	✓
Vertical Scaling	✗	Up	Up-Down

TABLE II. Comparison Table: Existing vs Autonomic Approach

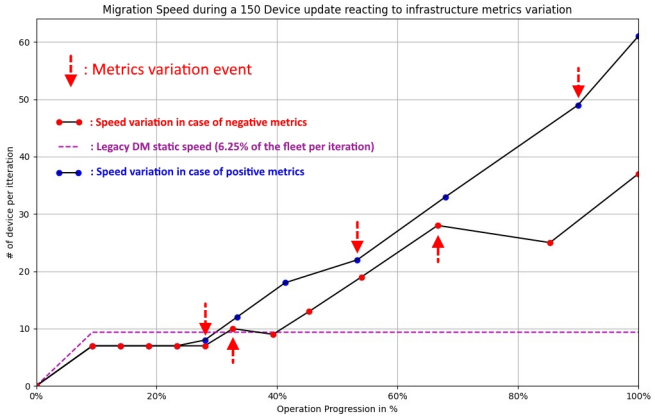

Figure 5. Autonomic DM: Reaction to DM Platform metric perturbations

reasonable speed that allows manual monitoring to be done and avoids call-center congestion). For comparison, we take the latter (Fastest rate) as reference regarding existing home DM solution execution speed.

B. Test Protocol

In order to test our system, we launch a predefined (150 for our tests) amount of DM Client that simulates our device fleet. Afterwards, we trigger a new firmware availability notification via the messaging queue. This leads Operation Generation & Target Identification loop to compute which device of the fleet will receive this firmware and starts the DM operation.

During each DM Operation Element execution its size is monitored and plotted (i.e., how much devices are to be processed). This allows to observe what decision how the autonomic system regulates the element size in normal iterations and perturbed ones. We introduce 2 types of perturbations (red arrows in Figure 5), positive and negative events.

- Positive variation of infrastructure response times and execution metrics.
- Negative variation of the aforementioned indicators.

If metrics improve compared to last iteration, accelerate update deployment by rising *Batch Size* (i.e., amount of devices processed in each iteration). Otherwise, it will either slow down (increasing amount of anomalies), stabilize (if little or no variation) or abort (if too many errors are detected).

For reference, Orange internal Home Device Management migration strategy's peak speed is represented in purple in Figure 5 (6.25% of the fleet per iteration).

In Figure 5 the number of devices per iteration is plotted as a function of overall progression. It details how our autonomic loops based approach reacts to metric variations.

Scenario 1 (Negative variation of metrics) colored in red in Figure 5 shows how autonomic management reacts to response time increase (red arrows in plot). Indeed, it slightly lowers execution speed for one iteration (as seen approx. at 35% 65%). Once infrastructure scaled up (therefore response time back to a lower value), the system increases batch size again in next iteration.

Scenario 2 (Positive variation of metrics) colored in blue in Figure 5 shows how our system manages improving metrics coming from DM Infrastructure at 27%, 55% and 90%. Slopes between these points and next ones is higher (therefore speed variation too). In fact, autonomic Management interpret lower response time (and error count) as a sign of resources availability at the IoT Platform level.

C. Discussion

Table II compares our approach with existing Orange Home DM Platform and industrial IoT Platforms, DM protocol support, execution speed adaptation, reaction to anomalies and vertical scaling.

While Orange Home DM and Industrial IoT rely on system administrators for operation launching and target identification, autonomic approach monitors new firmware or configuration availability by itself and also triggers update operations when a fleet is not up-to-date anymore (e.g., new device arrival). Both of current Home and IoT DM platforms respectively support single or few DM standard protocols. Autonomic Enhanced approach relies on underlying platform for connectivity. Two key features of autonomic management are speed regulation based on error & infrastructure metrics and anomaly awareness. These allow more accurate platform operation and avoid faulty configuration and firmware to be generalized creating service interruption. Finally, while Home DM do not offer vertical scaling [8], since industrial IoT platform are designed to run on Cloud Infrastructure such as Microsoft and Amazon ones, they allow up and down scaling. However, since they rely on the infrastructure beneath them to scale, if not hosted in an unrestricted elastic cloud environment or in case of limited resources available, no scaling is possible. Autonomic Enhanced DM enables full vertical scaling in the DM system directly by regulating execution speed.

Our tests are realized in a local server with some great compute capabilities. Yet for experimentation accuracy, real life testing on IoT platforms such as Azure, AWS or IBM could make experimental results richer. This requires to have access to source code and instances of them in order to modify their behavior and integrate an autonomic management layer on top. While Vertical Scalability is addressed in this paper, horizontal scalability is currently in investigation. It is required for massive (Millions, Billions) amount of IoT device management. Last, our proposal protocol compatibility is currently enabled via platform protocol support. Our design was thought to be

protocol agnostic by design. With middleware acting a proxy translating ourgoing orders from autonomic management to protocol specific commands this issue should be resolved.

V. Conclusion & Perspectives

In this paper, we address three of identified IoT DM challenges. **Heterogeneity**, **dynamicity** and **scalability** of devices makes existing Home, MDM and workstation DM solutions used at Orange and their industrial IoT counterparts unsuitable for IoT DM. This is due to their manual operation, static execution speed and lack of impact detection (e.g., device errors, infrastructure overload).

We propose a novel autonomic approach for IoT device management. It relies on:

- A Coordinated multi-loop architecture for IoT DM
- An **Operation Generation & Target Identification** loop that automatically targets suitable devices.
- A **Decomposition, Enforcement and Tracking** loop that executes DM operations and monitors devices and infrastructure.
- A Speed Regulation loop that regulates **Decomposition, Enforcement and Tracking** speed according to anomalies and infrastructure load.
- A Proof of Concept for experimental validation.

In terms of perspectives, we aim to validate autonomic target identification by interfacing our autonomic IoT DM managers to Orange LiveObjects [23] cloud platform. We have ongoing work regarding IoT challenges that have not been addressed in this paper (i.e., Interoperability and Scalability). Thus, we are investigating our proposal's scaling capability through distribution at the edge of the network, that is a key requirement for massive IoT device management. This will allow numerous IoT devices management via **horizontal scalability** in addition to vertical scalability detailed in this paper [8]. We are also exploring several millions of devices simulation on Grid5000 infrastructure [30].

References

[1] "11 best practices for mobile device management (mdm), ibm security thought leadership white paper." [Online]. Available: https://www.ibm.com/downloads/cas/VENWY8OG

[2] "Broadband forum." [Online]. Available: https://www.broadband-forum.org/

[3] N. Ayeb, E. Rutten, S. Bolle, T. Coupaye, and M. Douet, "Towards an autonomic and distributed device management for the internet of things," in 2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W), Jun 2019, p. 246–248.

[4] "Google home firmware update is bricking some units." [Online]. Available: https://9to5google.com/2019/10/24/google-home-firmware-brick/

[5] "Rbc analyst says 52 million google home devices sold." [Online]. Available: https://voicebot.ai/2018/12/24/rbc-analyst-says-52-million-google-home-devices-sold-to-date/

[6] "Hpe support." [Online]. Available: https://support.hpe.com/hpesc/public/docDisplay?docLocale=en_US&docId=a00097382en_us

[7] "Dell emc support." [Online]. Available: https://www.dell.com/support/home/fr/fr/frbsdt1/drivers/driversdetails?driverid=8h6hj&oscode=w12r2

[8] M. Michael, J. E. Moreira, D. Shiloach, and R. W. Wisniewski, "Scale-up x scale-out: A case study using nutch/lucene," in 2007 IEEE International Parallel and Distributed Processing Symposium, 2007, pp. 1–8.

[9] N. Gligorić, S. Krčo, D. Drajić, S. Jokić, and B. Jakovljević, "M2m device management in lte networks," in 2011 19th Telecommunications Forum (TELFOR) Proceedings of Papers, Nov 2011, p. 414–417.

[10] A. A. Corici, R. Shrestha, G. Carella, A. Elmangoush, R. Steinke, and T. Magedanz, "A solution for provisioning reliable m2m infrastructures using sdn and device management," in 2015 3rd International Conference on Information and Communication Technology (ICoICT), May 2015, p. 81–86.

[11] I. Danila, R. Dobrescu, D. Popescu, R. Marcu, and L. Ichim, "M2m service platforms and device management," in 2015 9th International Symposium on Advanced Topics in Electrical Engineering (ATEE), May 2015, p. 67–72.

[12] M. N. Islam and S. Kundu, "Remote configuration of integrated circuit features and firmware management via smart contract," in 2019 IEEE International Conference on Blockchain (Blockchain), Jul 2019, p. 325–331.

[13] H. Gupta and P. C. van Oorschot, "Onboarding and software update architecture for iot devices," in 2019 17th International Conference on Privacy, Security and Trust (PST), Aug 2019, p. 1–11.

[14] S. Choi and J.-H. Lee, "Blockchain-based distributed firmware update architecture for iot devices," IEEE Access, vol. 8, 2020, p. 37518–37525.

[15] S. Rao, D. Chendanda, C. Deshpande, and V. Lakkundi, "Implementing lwm2m in constrained iot devices," in 2015 IEEE Conference on Wireless Sensors (ICWiSe), Aug 2015, p. 52–57.

[16] M. Ha and T. Lindh, "Enabling dynamic and lightweight management of distributed bluetooth low energy devices," in 2018 International Conference on Computing, Networking and Communications (ICNC), Mar 2018, p. 620–624.

[17] A. Karaagac, M. VanEeghem, J. Rossev, B. Moons, E. DePoorter, and J. Hoebeke, "Extensions to lwm2m for intermittent connectivity and improved efficiency," in 2018 IEEE Conference on Standards for Communications and Networking (CSCN), Oct 2018, p. 1–6.

[18] "Google kills android distribution numbers on the web, but we've got you covered." [Online]. Available: https://9to5google.com/2020/04/10/google-kills-android-distribution-numbers-web/

[19] "New android 11 devices will support virtual a/b for seamless updates." [Online]. Available: https://www.xda-developers.com/google-virtual-ab-seamless-updates-android-11/

[20] "Broadband forum cpe wan management protocol (cwmp) data models." [Online]. Available: https://cwmp-data-models.broadband-forum.org//

[21] "Genieacs." [Online]. Available: https://github.com/genieacs/genieacs

[22] "Freeacs." [Online]. Available: https://github.com/freeacs/freeacs

[23] "Orange live objects." [Online]. Available: https://liveobjects.orange-business.com/#/liveobjects

[24] "Amazon aws iot device management." [Online]. Available: https://aws.amazon.com/iot-device-management

[25] "Azure iot hub." [Online]. Available: https://azure.microsoft.com/en-us/services/iot-hub/

[26] "Software updates over the air - bosch iot suite." [Online]. Available: https://www.bosch-iot-suite.com/software-updates-over-the-air/

[27] "Oma specifications." [Online]. Available: http://openmobilealliance.org/wp/index.html

[28] "User services platform (usp)." [Online]. Available: https://usp-data-models.broadband-forum.org/

[29] "Eclipse leshan." [Online]. Available: https://github.com/eclipse/leshan/

[30] "Grid'5000: A large-scale and flexible testbed for experiment-driven research in all areas of computer science." [Online]. Available: https://www.grid5000.fr/w/Grid5000:Home