Towards an Autonomic and Distributed Device Management for the Internet of Things

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

Towards an autonomic and distributed device management for the Internet of Things

Neil Ayeb
Orange
Meylan, France
firstname.lastname@orange.com

Academic Supervisor:
Eric Rutten
INRIA / LIG: Ctrl-A
Montbonnot-Saint-Martin, France
firstname.lastname@inria.fr

Industrial Supervisors:
Sebastien Bolle, Thierry Coupaye, Marc Douet
Orange
Meylan, France
firstname.lastname@orange.com

***Abstract*—Device Management (DM) is currently industrially deployed for LAN devices, phones and workstation management. Internet of Things (IoT) devices are massive, dynamic, heterogeneous, and inter-operable. Existing solutions are not suitable for IoT management. This doctoral research in an industrial environment addresses these limitations with a novel autonomic and distributed approach for the DM.**

***Index Terms*—autonomic computing, device management, firmware update, configuration, distribution, internet of things.**

I. Context and motivation

A. Motivation: DM for the IoT

ABIResearch affirm that Device Management (DM) of the Internet of Things (IoT) will grow to US$20.5 Billion by 2023 [1]. DM consists of remote and potentially massive operations on a fleet of deployed devices (e.g., home gateways, Set-top-boxes, IoT gateways, sensors ...). It is currently massively used by Internet Service Providers (ISP) for LAN devices management and industrials for smartphones and workstations management.

Main features of DM are the following

- Provisioning: initial and 'in-life' setup and configuration
- Monitoring: probe data pushing
- Assistance: remote troubleshoot
- Maintenance: firmware updates

IoT relies on massive deployment of connected devices. This implies several constraints such as high number of devices, heterogeneity their types and capabilities and dynamic environments. Challenges emerge from IoT DM:

- Heterogeneity: Current use-cases are targeting a *small* and *fixed* set of device types. For instance, ISPs usually manage consumer premises equipments (CPE) such as home gateways and set-top-boxes. IoT implies that numerous device types will be dynamically managed by multiple actors (owners or users).
- Dynamicity: Device Management is commonly designed to be static and ad-hoc (i.e., static operation-rules, behavior, device targets). IoT requires a more state-aware DM (e.g., device internal and external environments).
- Interoperability: Usual DM supposes that for each device type and its management platform there is no need for external communication or coordination. This assumption should be reconsidered in an IoT context with the proliferation of over the top (OTT) services deployed on non-fully managed devices.
- Scalability: Current DM is based on centralized designs that are manually configured [2] [3] [4] [5]. IoT's high number of devices [6] implies a significant increase of monitoring data sent to the DM platform. Thus distributing these will allow more efficient network usage and lower reaction times to errors or alerts.

B. Limitations

1) Classical DM solutions limitations: Administrators, industrials, and service providers resort to various DM solutions depending on their device target. ISP's and LAN device manufacturers tend to use TR-069 based DM platforms such as GenieACS [7], FreeACS [8] or internally developed solutions.

Industrials utilizes Mobile Device Management solutions [3] [4] and workstation administration tools to remotely manage their device fleets (phones, tablets, servers, workstations...). For instance, Microsoft System Configuration Manager [5] allows the aforementioned devices to be managed via a single, yet closed, non-standardized and mono-actor platform.

These solutions rely on manual configuration and driving by a system administrator. Experience reports from industrials shows that even with a small and fixed set of device types, these solutions **are** complex to operate.

2) Industrial IoT platforms DM limitations: Commercial IoT platforms such as AWS IoT [9], Azure IoT Hub [10], or Orange Live Objects [11] claim to be able to execute an operation on a massive float of devices with conditions on their state (values of their data-model). Actions targets and conditions are still managed by the administrator via a platform-wide dashboard. To the extent of our knowledge, these commercial platforms do not include an operation model allowing granular tracking of the execution. Such mechanisms would help (in case of operation failure) identifying what step failed and aborting operations on pending and on-going devices.

Current industrial solutions are centralized and designed to manage a small and static set of devices. These solutions are driven by a single actor's administrator. With IoT's challenges (i.e., heterogeneity, dynamicity, interoperability, scalability)

complexity will be challenging system administrators making them less efficient in DM solutions configuration and operation. Moreover, centralized designs are not adequate to manage massive remote DM operations and their granular tracking.

II. Objectives & Contribution

A. Objectives

1) Automation: In order to tackle complexity challenge, automation is used as a means of abstracting a system's granular functioning details and system initiated action triggering. Autonomic Computing (AC) is defined by [12] as the self-management capabilities of a system. AC purpose is addressing the increasing complexity of the system administration. An autonomic system can react and adapt to external and internal changes by reconfiguring itself. This behavior has been traditionally within the responsibilities of the system administrator. AC allows to reduce the complexity of system management via less interactions with the administrator and complexity abstraction. The autonomic behavior of a system is enabled via autonomic managers dispatching and coordination [13] [14].

2) Distribution: In order to tackle the scalability issues of the static and centralized design of currently used DM solutions, it is aimed to dynamically distribute the platform and its managing system. Such design allows more efficient usage of the infrastructure (network, computing and memory capabilities). For instance, monitoring data will be pushed to a geographically close node instead of the existing cloud hosted storage. Besides, having parts of the DM platform closer to managed devices allow better reactivity, higher privacy, and enhanced troubleshooting.

B. Contribution

1) Originality: DM Platforms are not designed with the IoT in mind. Main solutions target LAN devices, phones or workstations. Even though IoT Platforms (e.g., Azure, AWS, Live Objects) include a DM module, they are centralized and manually configured. Distribution for the IoT is mainly about data and services are rarely tackles DM challenges.

2) Adequacy: In order to address the complexity brought by heterogeneity, automation seems adequate. For dynamicity, loop-based control and automation is identified. Interoperability is tackled with model-based aspect of DM automation. Scalability is treated via distribution.

III. Methodology & Early Results

A. Methodology

1) Operation automation and tracking: This loops objective is to keep the fleet up to date, potentially considering interdependencies between devices. It consists of monitoring the device fleet states (i.e., internal and external environments), then computing if DM operations are required. An administrator still can manually trigger these operations. Then it reschedules or modifies the pending or on-going operations if needed. Finally, it keeps tracking the execution of the pushed operations for error management policies enforcement (e.g., stop applying a faulty operation, rollback, alert triggering). The control loop can be operating at a fixed frequency (using parameters such as: amount of on-going operations, fleet size, device type...) or event-triggered.

Via this autonomic manager, the DM platform is going towards self-operation and self-healing. Indeed, beside the traditional configuration by the administrator, the platform can trigger DM operations in an autonomous manner. It is also able to track operations and react to errors unlike existing designs where error detection and reaction is not fully automated and tracking manually realized by the administrator.

2) DM platform and control resource usage regulation: This loops objective is to scale up and down depending on high level objectives and load. It monitors the infrastructure used to operate the DM platform and its autonomic managers (e.g., network load, hops, memory and CPU usage). Based on KPIs received from the platform administrator, reconfiguration will be considered and if needed applied. Actual enforcement of the reconfiguration plan is executed via DM servers (de)instantiation or control loops distribution or centralization. For this manager we will consider model-based appraoches, related to application of control theory [15].

3) Multi-loop coordination and distribution: The two identified managers need interaction and coordination either via a higher global manager (hierarchical) or via direct connection (e.g., master/slave, local coordination, shared elements) [13] [14] [16].

B. Early results

A first proof-of-concept that has been implemented and is composed of the following entities:

- Two DM Clients (representing two devices) with different battery level and firmware version.
- A single DM Server that manages both devices.
- An autonomic manager which manages the DM platform.

For the implementation, we used *Python* [17] for the autonomic loop design and the open-source project Eclipse Leshan [18] that we modified for the client and the server.

Leshan project is a Java implementation of an OMA LightWeight M2M [19] DM Protocol, client and server (among other components). The LWM2M Client has been modified to simulate a *charging* device and an *always plugged-in* device.

A device that is compliant (state-wise) to a pushed operation condition will be updated, while the other will stay on a 'pending' state until compliance or operation expiry.

IV. Future Work

Next steps consists of multiple decentralized DM instance and distributed control and current loop elaboration and interaction. Besides, modeling work is currently on-going to achieve model-based control [15]. This will allow interoperability, granular DM operation tracking and troubleshooting, and scalability. Distribution criteria (e.g., geoloc, topology, load...) will also be studied.

REFERENCES

[1] "IoT Device Management Revenues to Climb to US$20.5 Billion by 2023". Available: https://www.abiresearch.com/press/iot-device-management-revenues-climb-us205-billion-2023/ [Accessed: 20-Feb-2019]

[2] Broadband Forum, "CPE WAN Management Protocol, Issue: 1 Amendment 6,(Technical Report 69)", March 2018.

[3] "Citrix XenMobile" [Online]. Available: https://docs.citrix.com/en-us/xenmobile.html [Accessed: 28-Feb-2019].

[4] "VMware AirWatch" [Online]. Available: https://docs.vmware.com/en/VMware-AirWatch/index.html [Accessed: 27-Feb-2019].

[5] "Microsoft System Center Configuration Manager" [Online]. Available: https://www.microsoft.com/en-us/cloud-platform/system-center-configuration-manager [Accessed: 10-Mar-2019].

[6] Gartner, "https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf," [Accessed: 20-Feb-2019]

[7] "GenieACS" [Online]. Available: https://github.com/genieacs/genieacs/ [Accessed: 28-Feb-2019].

[8] "Freeacs" [Online]. Available: https://github.com/freeacs/freeacs [Accessed: 27-Feb-2019].

[9] "AWS IoT Device Management Documentation." [Online]. Available: https://docs.aws.amazon.com/iot-device-management/index.html. [Accessed: 28-Feb-2019].

[10] "Azure IoT Hub Documentation - Tutorials, API Reference." [Online]. Available: https://docs.microsoft.com/en-us/azure/iot-hub/. [Accessed: 28-Feb-2019].

[11] "Orange Live Objects." [Online]. Available: https://liveobjects.orange-business.com/ [Accessed: 28-Feb-2019].

[12] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," Computer, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[13] D. Weyns, B. Schmerl, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. Andersson, H. Giese, K. M. Göschka, "On Patterns for Decentralized Control in Self-Adaptive Systems," Software Engineering for Self-Adaptive Systems II, pp. 76–107, 2013.

[14] A. Al-Shishtawy, V. Vlassov, P. Brand, and S. Haridi, "A Design Methodology for Self-Management in Distributed Environments," in 2009 International Conference on Computational Science and Engineering, Vancouver, BC, Canada, 2009, pp. 430–436.

[15] M. Litoiu, M. Shaw, G. Tamura, N. M. Villegas, H. A. Müller, H. Giese, R. Rouvoy, E. Rutten, "What Can Control Theory Teach Us About Assurances in Self-Adaptive Software Systems?," Software Engineering for Self-Adaptive Systems III. Assurances, pp. 90–134, 2017.

[16] A. N. Sylla, M. Louvel, E. Rutten, and G. Delaval, "Design Framework for Reliable Multiple Autonomic Loops in Smart Environments," in 2017 International Conference on Cloud and Autonomic Computing (ICCAC), 2017, pp. 131–142.

[17] "Python Software Foundation." Python Language Reference, version 3.7. Available at http://www.python.org

[18] "Eclipse Leshan." [Online]. Available: https://github.com/eclipse/leshan [Accessed: 28-Feb-2019].

[19] "OMA Lightweight M2M." [Online]. Available: https://www.omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/ [Accessed: 28-Feb-2019].