Managing Connected Smart Objects

Alan McGibney(✉), Alejandro Esquiva Rodriguez, Oliva Brickley, and Susan Rea

Nimbus Centre, Rossa Avenue, Bishopstown, Cork, Ireland
{alan.mcgibney,oliva.brickley,susan.rea}@cit.ie,
a.esquiva-rodraguez@mycit.ie

Abstract. This paper provides an initial architecture specification of a management framework to address the challenges associated with the robustness and reliability of large scale IoT systems, specifically through mechanisms for orchestration of resources for reliability and dependability supported by IoT functional virtualization.

Keywords: IoT management · Architecture · Software defined networking

1 Introduction

The Internet of Things (IoT) domain continues to grow rapidly and the numbers of devices, interfaces, operating environments, applications and services has exploded making the need for integration and management of intelligent devices and data more critical if efficiencies of scale are to be achieved within IoT ecosystems. The IERC in [1] explore three macro challenges that have been identified in relation to the evolution of IoT - namely: billions of devices, IOT management for robustness and reliability, and thirdly intelligent reasoning over IoT data. Forecasts tell us that by 2020 there will be in excess of 50 billion devices connected to the Internet generating massive volumes of data that must be delivered on time, in the correct format and to the appropriate end-user in order to deliver sustainable services, this drives the need for robust IoT management and the ability to transform raw data streams into actionable knowledge. This paper focuses on the IoT management for robustness and reliability macro challenge and specifically the sub-challenges of (i) orchestration of resources for reliability and dependability, and (ii) IoT functional virtualization. Resource orchestration relates to the ability to evaluate dependencies between sensing requirements, networking and physical resources and their relationship to QoE and reliability and the overlaying application. Functional virtualization offers the ability to decouple applications from the underlying physical infrastructure and can support multi-tenant applications. This paper will provide the initial specification of an architecture (utilizing the IoT-A reference architecture as a basis) to support seamless interaction between IoT applications and connected smart objects (things, people, and products) through the realization of a framework, based on the software defined networking (SDN) principles, which bridges the digital and physical system components. This will result in the specification of reference architecture for infrastructure (embedded and mobile devices) management within IoT ecosystems where the orchestration of resources and virtualization is addressed through SDN partitioning using physical and virtual partitioning to create a service management framework for sustainable service delivery.

2 Current Approaches to Smart Object Management

The integration of smart objects into IoT platforms typically focus on providing a data-centric mechanism for IoT services. This involves data being published remotely (either directly or via an Internet bridge), stored in a data warehouse and published to third party cloud platforms. This supports the development of content rich applications; however the impact these applications have on the underlying infrastructure is only loosely coupled. With the upsurge of connected devices, there is an ever increasing need to provide a set of autonomous management functions that integrate configuration, operation, administration, security, and maintenance of all elements of the IoT network. The dynamic grouping and autonomous management of smart objects expected by IoT applications means that traditional network management is not directly transferable in cases where manual configuration and tuning is often required by network administrators.

2.1 Software Defined Networking for IoT Infrastructures

SDN brings with it the opportunity to separate the control plane from the data plane, centralize the logical network control and abstract the underlying physical infrastructure from the applications and services being executed in the network [2]. The control plane is the core element in a SDN infrastructure and is defined as a set of software controllers that provide the network forwarding function and lies between the physical devices at one end and the applications/services at the other. Abstracting the entire underlying infrastructure in virtual entities makes it possible to create different network applications which accomplish the needs and requirements of different IoT applications, using the same infrastructure. Different IoT marketplaces have been appearing over recent years, which use virtualization of the network and the virtualization of the smart objects to provide end-to-end services without considering the characteristics or state of the network and their components. From the end-user view, the underlying infrastructure should be considered as an autonomous system which works as expected and is managed by itself. The translation of application requirements to the underlying infrastructure can be achieved by the generation of policies, which will provide a set of rules to meet application specifications. In terms of SDN this task is handling by the North Bound API. There are several policy-based languages (Procera, Frenetic, FML and Nettle) which build a policy layer on top of existing controllers and are responsible for converting high-level policies to flow constraints that are be used by the controller. However, these policy-based languages are standalone and provide compatibility only with the South Bound API standard OpenFlow. A common alternative for the North Bound API are RESTFul APIs which are simple and straightforward for developers, but often do not support the metadata necessary for programing automation [3]. Currently, for the North Bound API there is no standard approach defined, and the existing solutions do not accommodate the needs of general IoT ecosystems resulting in static, application specific APIs. In enterprise networking SDN has been promoted as a solution to manage large-scale distributed networks and brings with it scalability and reliability making it an attractive solution to manage, coordinate and optimize the use of resources in diverse IoT infrastructures. Key performance indicators at the control plane are managing scalability and guaranteeing reliability and to do this the deployment of multiple distributed controllers is required. Control partitioning is driven by horizontal or vertical partitioning in order to subdivide the physical infrastructure into multiple domains. Hand-in-hand with this is the need to identify where to place controllers and how many controllers are required when dimensioning the network. Approaches to distributed control partitioning include Onix [4], DevoFlow [5], HyperFlow [6]. These SDN technologies are designed based on assigning fixed partitions over a rigid infrastructure with static controller assignment per partition where the network infrastructure is considered to be composed of typically wired switches, routers and high end servers. However, in IoT infrastructures such networking infrastructure will not be the norm and physical devices will be dispersed over a significantly more loosely coupled heterogeneous environment. The complementary problem of controller placement and quantity has been explored in [7, 8] and their findings surmise that no generalization can be drawn on the numbers and placement of controllers that is applicable to all networks. Placement depends on the network topography and performance metrics defined by service level agreements (SLAs) that need to be met based on delay, bandwidth, and reliability for example. SDN Controller communications is driven by one of three communications interfaces, namely the northbound, southbound and east-west bound interfaces. OpenFlow is the protocol that is most widely accepted as de facto when implementing SDN solutions for the southbound communications interface and on the northbound interface REST APIs are proving popular choice among vendors. The east-west bound interface while being accepted as being needed to support inter-controller communications between partitions or federations of devices is not currently supported by any standard or established protocol and is as such the most immature among the available interfaces in SDN implementations but is critical in order to manage federations of SDN domains.

3 Proposed Management Framework

IoT-A [9] was developed to define the basic IoT domain concepts and the functional components of the reference architecture. Furthermore, IoT-A encapsulates a set of models to establish a common understanding of the IoT domain such as a Domain Model to identify entities, resources and services that are important actors of the IoT scenario, and an Information Model which specifies how the information will be modelled. Utilizing a direct instance of the IoT-A reference architecture as a basis, we have created the initial specification of an architecture to support the management of connected smart objects. An abstract representation of the framework is provided by the functional model that defines IoT-A. The functional model is composed of several functionality groups (FGs) which provide the functionalities for interacting with the instances of these concepts or managing the information related to the concepts.

Figure 1 shows our functional model instanced by the IoT-A. Each FG is composed of different functional modules (FMs).

The *Management FG* encapsulates the control and maintenance modules of the framework, providing tools to assess the performance of smart object networks in terms of resources constraints, throughput, connectivity etc. The *Service Organisation FG* maintains a domain registry including a semantic model of smart object(s) and partitions. The *IoT Process Management* provides the network operating system that is responsible for generating, mediating and invoking service policies between applications and smart objects. The *Virtual Entity* and *IoT Service FGs* provide capabilities to represent the current state of resources, conflict resolution between service requirements and a set of services to monitor and maintain system stability. The *Communication FG* provides an abstraction from the various interfaces required to form a communications backbone for the components of the framework. This component is logically located in the smart object, with the aim to provide common APIs for

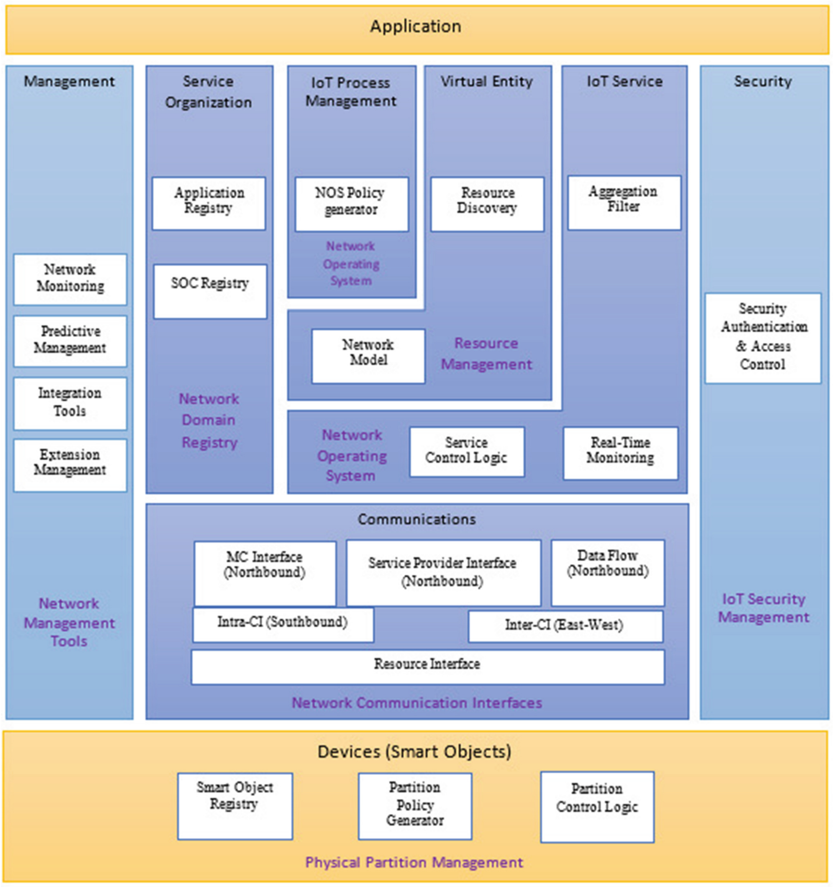

Fig. 1. A functional view of smart object management framework.

exploiting resources, manage network services and handle inconsistence issues. The *Device FG* is not included in the mapping of functional Unified Requirements of the IoT-A while it needs to be considered when devising a concrete IoT system. However from an infrastructure management perspective, the *Device FG* plays an important role, as the devices need to host common functional blocks to enable autonomous management between smart objects. From an implementation perspective the type of device will dictate the level of sophistication of the controllers deployed on them. A key aspect to the proposed management framework is to ability to dynamically partition the infrastructure to meet IoT application needs. Figure 2 provides a high level functional view of the proposed partitioning mechanism. It is composed of a number of functional blocks that from one side allow for the dynamic partitioning of physical infrastructures and from another derive a virtual overlay to service IoT applications.

A number of context-aware smart controllers are required to coordinate and optimize the use of physical resources. A key component is the development of a Master Controller (MC) which is responsible for the discovery, classification and registration of smart objects within the management framework. Thus, the MC will act as the initial mediator between applications and the supporting physical infrastructure. The MC will utilize existing approaches for the classification, registration and discovery of smart objects. It is assumed that each smart object has a remotely configured Smart Object Controller (SOC). This concept will enable smart object bootstrapping where they connect and register with the resource repository. To realize a SOC requires the following base functionality, RESTful service interface to consume and process RESTful requests, object interface (for control of physical devices), information model and embedded management functionality. The use of the reference framework specification will ensure implementation is consistent however the execution environment may result

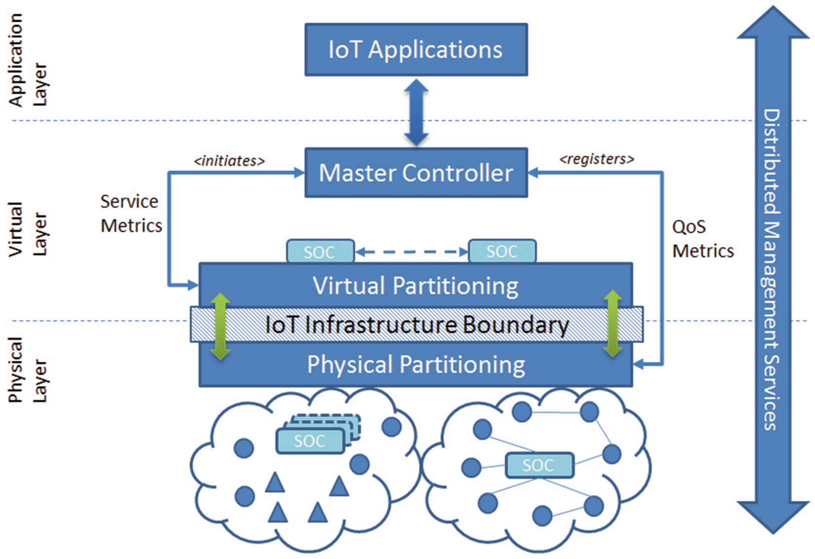

Fig. 2. IoT infrastructure partitioning

in a number of SOC types. From an IoT system management perspective, EU FP7 project *iCore* provides a base framework for the representation, registration, composition and discovery of virtual entities to support multiple IoT applications over the same infrastructure. Similarly EU FP7 project *OpenIoT* provides mechanisms for context-aware ranking of smart objects to establish relevance to application needs; it is acknowledged that many middleware platforms provide basic search capabilities (proximity and data type) focus and as such should be leveraged by the proposed framework.

To distribute the management functionality requires the creation of localized controllers that can proactively partition the physical infrastructure to meet varying QoS demands. The formation of these **physical partitions** in highly-dynamic IoT environments is analogous to clustering mechanisms used in MANETs. Within this domain a number of works have been proposed for clustering, each using various factors to drive the election of a device as a cluster head (CH). The factors utilized include the degree of the nodes (the number of neighbors that a node has), node mobility and node energy. In addition there exist approaches that target the combination of these factors as weighted metrics to optimize the selection of the cluster head. The challenge is to maintain the stability of the network once the cluster head has been elected, it is impacted by mobility of devices, resource depletion, connectivity etc. This challenge is exacerbated further in IoT systems where a number of virtual overlays driven by service level agreements are formed and rely on the stability of the underlying physical infrastructure. Therefore it is proposed to extend these metrics to incorporate the current context (how the smart objects are being used, what they are being used for, and the resources available, smart object/service priority and current SLAs being supported).

Virtual Partitioning will create an elastic overlay that collapses or expands depending on the current context and workload. Unlike existing SDN implementations, controllers will be dynamically created and deployed based current context including new and existing service requests and will manage smart objects within specific regions of the infrastructures. Neither enterprise wired SDN solutions such as OniX, DevoFloW, FlowVisor or wireless network SDN approaches such as TinySDN, Sensor OpenFlow, SDWN, UbiFlow, CellSDN and OpenWireless consider dynamic controller creation, assignment or deployment. These solutions rely on proactive partitioning and controller assignment, where all controllers are assumed to be statically deployed on permanent devices for the duration of the network with fixed end devices. We argue that in order to manage connected smart objects in the context of IoT relying on the need for fixed infrastructure to manage scalability is not appropriate and clustering devices must be reactive based on current context. Smart Object Controllers (SOC) are software modules that are assigned and deployed on appropriate smart objects at specific instances of time and can be moved, split or consumed at will, driven by the IoT infrastructure workload. Virtual partitions can expand and consume other partitions with the controller taking over the management of the consumed smart objects. Likewise partitions can collapse when controllers experience overload and need to shed resources and new controllers and will be dynamically created to manage the resources released. To support the collapsing and expanding of federations it is proposed to develop an east-west bound controller management protocol to support inter-controller communications and will leverage existing protocols such as OpenFlow, BGP, EIGRP and will be used for partition maintenance and stabilization. The primary focus for this framework is context driven partitioning and dynamic controller creation/assignment rather than flow management. To enable flow management, load balancing, mobility and handover management the framework will adopt concepts from SDN wireless solutions such as UbiFloW (which has been designed for urban scale software defined IoT networking and uses distributed hash tables to support mobility and network calculus for flow management), CellSDN, which targets policies for cellular applications that are driven by subscriber needs, instead of physical locations, Sensor Flow targets SDN over WSNs and OpenWireless which looks at handover for high rate video stream between networks. A critical consideration is the impact the formation of virtual partitions will have on the physical infrastructure; this is represented as the IoT infrastructure boundary in Fig. 2, this requires cross-layer collaboration among smart object controllers to ensure firstly, the physical infrastructure can support IoT services and secondly, the virtual partitioning of smart objects does not to detrimentally impact the stability of the supporting infrastructure.

This paper provides a functional specification of a management framework based on the IoT-A reference architecture that aims to ensure the impact of multi-tenant IoT applications has on the stability and robustness of the supporting physical infrastructure is taken into account during development. Future work includes the realization of a concrete instance of the reference architecture and the development of novel methodologies using concepts of SDN to manage the boundary between the physical and virtual partitions that support future dynamic and scalable IoT applications.

References

1. Vermesan, O., Friss, P.: Building the Hyperconnected Society - IoT Research and Innovation Value Chains, Ecosystems and Markets. River Publishers, Aalborg (2015). ISBN 978-87-93237-99-5
2. Jarraya, Y., Madi, T., Debbabi, M.: A survey and a layered taxonomy of software-defined networking. IEEE Commun. Surv. Tutor. **16**(4), 1955–1980 (2014)
3. Dmitry, N., Sneps-Sneppe, M.: Metadata in SDN API (2015). arXiv preprint: arXiv:1503.06630
4. Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., Shenker, S.: Onix: a distributed control platform for large-scale production networks. In: USENIX OSDI (2010)
5. Curtis, A.R., et al.: Devoflow: scaling flow management for high-performance networks. In: ACM SIGCOMM (2011)
6. Tootoonchian, A., Ganjali, Y.: HyperFlow: a distributed control plane for OpenFlow. In: INM/WREN (2010)
7. Heller, B., Sherwood, R., McKeown, N.: The controller placement problem. In: HotSDN (2012)
8. Hu, Y., et al.: On the placement of controllers in software-defined networks. China Univ. Posts Telecommun. **19**(S2), 92–171 (2012)
9. Bassi, A., et al.: Enabling Things to Talk, Designing IoT Solutions With the IoT Architectural Reference Model. Springer, Heidelberg (2013). ISBN 978-3-642-40402-3