Computers and Electrical Engineering 68 (2018) 570–580

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Democratization of runtime verification for internet of things☆

Koray Incki*, Ismail Ari

Department of Computer Science, Ozyegin University, Cekmekoy, Istanbul 34794, Turkey

ARTICLE INFO

Keywords:
Internet of things
Model-based testing
Runtime verification
Complex-event processing
Model-to-text transformation
Edge computing

ABSTRACT

Internet of Things (IoT) devices have gained more prevalence in ambient assisted living (AAL) systems. Reliability of AAL systems is critical especially in assuring the safety and well-being of elderly people. Runtime verification (RV) is described as checking whether the observed behavior of a system conforms to its expected behavior. RV techniques generally involve heavy formal methods; thus, it is poorly utilized in the industry. Therefore, we propose a democratization of RV for IoT systems by presenting a model-based testing (MBT) approach. To enable modeling expected behaviors of an IoT system, we first describe an extension to a UML profile. Then, we capture the expected behavior of an interaction that is modeled on a Sequence Diagram (SD). Later, the expected behaviors are translated into runtime monitor statements expressed in Event-Processing Language (EPL), which are executed at the edge of the IoT network. We further demonstrate our contributions on a sample AAL system.

1. Introduction

IoT deployments are widespread in a variety of domains, enabling a more connected computing paradigm through *smart things* [1,2]. Early IoT adoptions were mostly designed to collect data from environment, and convey it to other processing units for intelligence gathering purposes. The increasing computing capacity of *things* enabled equipping them with full-fledged operating systems and special-purpose software. Such improvements in IoT systems will allow us to deploy certain processing tasks at the edge of an IoT network, thereby providing expedited data processing without compromising data integrity and security. An edge computing solution aims to provide seamless integration between IoT and cloud computing by using the processing power of gateway devices at the edge of a network to filter, pre-process, aggregate or score IoT data. Edge computing solutions utilize the power and flexibility of cloud services to run complex analytics on all data and, in a feedback loop, support decisions and actions about and on the physical world.

A majority of research on RV utilizes formal methods and logic programming such as Linear Temporal Logic (LTL/T-LTL) [3] for specifying runtime monitors, that's why it has been considered intimidating by practitioners for more than 50 years. However, RV is a fundamental verification method that increases the user's confidence on a system under test (SUT), by exerting certain monitoring techniques on execution traces of SUT. Considering the fact that an IoT system of systems (SoS) may be composed of hundreds or thousands of devices that may be manufactured by a multitude of manufacturers, the reliability of such a large scale system could be further assured by leveraging a *feasible* RV solution. RV complements the reliability of IoT systems by black-box testing approach, by using a black-box testing, which is one of the most appropriate testing approaches for such heterogeneous large-scale systems [3].

IoT systems are increasingly deployed with service-oriented architecture (SOA) principles, due to its ease of implementation and vast community adoption [4]. Thus, RV solutions that favor service-centric [3] nature of those systems are more feasible. In this

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. M. M. Hassan.

* Corresponding author.
E-mail address: koray.incki@ozu.edu.tr (K. Incki).

https://doi.org/10.1016/j.compeleceng.2018.05.007
Received 31 January 2018; Received in revised form 6 May 2018; Accepted 7 May 2018

study, we extend our research on event-based RV of IoT systems as described in our previous works [5–7]. In [5], we presented a novel contribution of representing IoT systems in terms of events that occur as a result of *send events* of messages in the system. In [6], we demonstrated how MBT can be utilized for interoperability testing of IoT systems at runtime. Interoperability is often defined as the ability of two or more systems interacting through provided interfaces of collaborating parties ([8]). In emerging domains such as IoT, special purpose application layer protocols are proposed to govern the interoperability issues. Constrained-application protocol (CoAP) [9] is developed with best practices in the industry, which specifies the interaction model and messaging formats between IoT endpoints. Its fundamental messaging model inherits basic principles of RESTful API model in [10]. In [7], we extended our contribution in [5] to involve context-based event processing.

As described by our study on RV of IoT in [5], CoAP-based IoT systems can be expressed in terms of events occurring as message exchanges. The service-centric messaging model of CoAP facilitates RV. We presented a proof of concept implementation of MBT-based RV for IoT systems in [6]; however, the work fails to define a deployment model that supports practical applications. In this study, we extend our efforts in representing basic interoperability issues of IoT systems using CoAP. We improve our research by proposing a meta-model for CoAP by extending the UML profile, and utilizing the meta-model for automatically generating runtime monitors from behavioral models of IoT SUT.

In this paper, we also extend the framework proposed in [6] to incorporate MBT support for generating runtime monitors of SDs by using context-based complex-event processing (CEP). CEP techniques allow us to exert various transformation operations on simple events to yield complex events; which, in turn, support decision analysis processes. The proposed solution relies on collecting and aggregating runtime events from IoT devices and transferring them to a cloud-based CEP engine for further analysis. We complement our proposed solution by demonstrating it on a sample AAL system. AAL systems are frequently composed of collaborating units of sensors or actuators for improving vital conditions and supporting the daily activities of elderly people. We believe that the proposed solution framework is the first of its kind in proposing a democratization of RV in IoT systems.

The rest of the paper is structured as follows. Section 2 introduces fundamental techniques we utilize in developing our contributions; Section 3 motivates the research on verification of a sample AAL system; then a reference deployment architecture for verification at the edge of things is explained in Section 4. Section 5 demonstrates how to use the proposed architecture on the motivating example; and the results are discussed in Section 6. We provide succinct review of the relevant literature in Section 7.

2. Background

2.1. Constrained-application protocol

Proliferation of Internet-based utilities and technologies has permanently altered our lives for good. This phenomenon was reinforced by introduction of RESTful API guidelines [10], which have elevated SOA adoption. IoT SoS's also benefit from RESTful-like application layer protocol, namely CoAP that is developed by IETF working group CoRE (Constrained RESTful Environments). CoRE compiled a body of standards that govern application-layer interactions, most prominent of which is the Constrained-Application Protocol (CoAP). CoAP is developed on RESTful guidelines so as to allow developing system architectures through SOA principles. Although CoAP demonstrates HTTP-like interactions, it is specifically devised for resource-constrained devices with limited battery, low memory footprint and limited computation power.

CoAP dictates a client/server communication pattern as in other RESTful services. A CoAP client sends a Request Message to a CoAP server, which determines the required action with a special Method Code on a resource of the server. The resources hosted by a server are identified by URIs (Unified Resource Identifiers), just as services in HTTP. Should the server accomplish processing of the corresponding Request, it then sends a Response Message back to the originating Client with the proper Response Code. It's noteworthy that any CoAP device (entity) can behave as both a client and a server in machine-to-machine (M2M) interactions.

The messaging model of CoAP is an asynchronous interaction model. The messages are exchanged over UDP packets [9]. There are four different kinds of messages: Confirmable (CON), Non-confirmable (NON), Acknowledgment (ACK), and Reset (RST). Even though the communication is based on UDP, an optional reliability is provided with exponential back-off (i.e., if an ACK is not received for a CON message, the CON message is re-transmitted within exponentially increasing period of time). Thus, those four message kinds are exchanged in a Request/Response type of interaction. A Request can be conveyed via both a CON and NON message; and, the Response to a corresponding Request might be separately sent in a CON/NON message, as well as piggybacked in an ACK message.

Request and Response semantic of CoAP interactions enables us to picture the protocol layer as consisting of two intrinsic logical

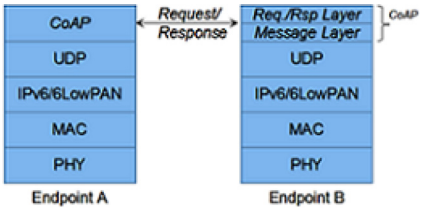

Fig. 1. CoAP OSI Layers.