Bluetooth Low Energy Peripherals and Edge-to-Cloud Data Assessment as a Service, with Docker, Node-RED, MQTT, Scala, Spark, Kafka, HDFS, and Android SDK: a demo*

Mirco Soderi[1 [0000−0003−3417−5741]] and John Gerard Breslin[1 [0000−0001−5790−050X]]

Data Science Institute, University of Galway, Galway, Ireland
{mirco.soderi,john.breslin}@universityofgalway.ie

Abstract. Reconfigurable manufacturing, or resilient manufacturing, has been a very active research field for more than two decades at today. It can be described as a complex of technologies that deal with cost-effective, quick reaction to context changes, including market changes. Remote and automated reconfiguration of both software and networking across the three layers of IoT computing (Edge, Fog, and Cloud) is part of that challenge, and the context for this work. In this manuscript, a demo is proposed where three components are involved: an IoT device, a mobile app, and a Cloud analytics server. A local data assessment takes place in the IoT device, through a K-Means Clustering. Values are sent to the mobile app via Bluetooth Low Energy (BLE), along with local assessments. The mobile app forwards the values to the Cloud analytics server for central assessment, and displays values, and assessments. All the software in the IoT device and on the Cloud is installed, configured, and run from remote through API calls, including the creation and implementation of the BLE service. All software but the mobile app is containerized. Graphical Integrated Development Environments (IDE), message brokers, event streaming servers, languages and frameworks for Cloud computing and analytics, and distributed file systems, are used.

Keywords: Reconfigurable Manufacturing · Resilient Manufacturing · Internet-of-Things (IoT) · Multi-Agent System · Self-organizing System · Microservice · API · Bluetooth Low Energy (BLE) · Message Broker · Events Streaming · Cloud Computing · Cloud Analytics · Cloud Storage · Containerization · Mobile App · Docker · Node-RED · MQTT · Scala · Spark · Kafka · HDFS · Android

* This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number SFI/16/RC/3918 (Confirm).

1 Introduction

In the early 2000s, reconfigurable manufacturing systems (RMS) were identified as the key to future manufacturing, a new paradigm designed for rapid adjustment of production capacity and functionality in response to new market conditions, which would have permitted to face the new challenges posed by globalization [4] [7]. Few years later, ElMaraghy [2] enunciated the main principles of the new discipline (modularity, integrability, flexibility, scalability, convertibility, and diagnosability), and Koren [3] highlighted how RMS differentiated from former Dedicated Manufacturing Lines (DML), and Flexible Manufacturing Systems (FMS). In 2010, Koren and Shpitalni [5] introduced a rigorous mathematical method for designing reconfigurable manufacturing systems. In late 2010s, Bortolini, Galizia and Mora [1] reviewed RMS application areas, key methodologies and tools, thus identifying five emerging research streams in reconfigurable manufacturing: (i) reconfigurability level assessment; (ii) analysis of RMS features; (iii) analysis of RMS performances; (iv) applied research and field applications, where this work falls; (v) reconfigurability toward Industry 4.0 [6] goals.

More recently, Morgan, Halton, Qiao, and Breslin [8] reviewed the research areas of RMS, Machine Control, and Machine Intelligence, with the aim of assessing the reconfigurable design and industry adoption, and of identifying the enabling present and future state technology, so establishing a vision for next-generation Industry 4.0 manufacturing machines. Soderi, Kamath, Morgan, and Breslin [12], proposed modular and reconfigurable Node-RED applications, named Service Nodes, to be run in Docker containers, and interconnected through MQTT brokers, for ubiquitous system integration as a Service in smart factories. Also, they proposed a containerized Scala + Spark server application for Cloud analytics, to be integrated into the system outlined in [12], by means of a Node-RED module to be loaded to the Service Nodes via API calls [13]. For verifying the versatility of the proposed framework, and in consideration of the importance of real-time Big Data visualization in most smart manufacturing scenarios, Soderi, Kamath, and Breslin [11], also started the building of an API-driven infinite cyber-screen for custom real-time display of Big Data streams, relying on the same building blocks and technologies that were proposed in [13] and extensively used in [10] for Big Data engineering and analytics. In end, Soderi and Breslin [9] proposed the Crazy Nodes, which are maximally flexible Service Nodes that expose APIs not only for loading pre-existing software modules from a targeted external library node, but also for performing arbitrary modifications on the node implementation, with immediate effect, and without any service disruption.

The paper is structured as follows. Reconfigurable manufacturing is introduced in Section 1. An overview of our proposed framework for ubiquitous, and remotely reconfigurable data communication, engineering, visualization, and analytics, is given in Section 2. The demo of Bluetooth Low Energy Peripherals and Edge-to-Cloud Data Assessment as a Service is presented in Section 3. Conclusions are drawn in Section 4.

2 Framework

The entry point for our proposed framework for ubiquitous, and remotely reconfigurable data communication, engineering, visualization, and analytics, is the **Network Factory**. It is distributed as a Docker image, publicly available on Docker Hub[1]. It is a Node-RED application that expose APIs for building, and managing at a high-level, Edge-to-Cloud computation networks. Each node is deployed as a Docker container. *Typical* nodes are: (i) Service Nodes, (ii) Crazy Nodes, (iii) BLE Nodes, (iv) AI Servers, (v) Transformation Libraries, (vi) Access Control List (ACL) API nodes for Service and Crazy Nodes, (vii) MQTT brokers, and (viii) ACL API nodes for MQTT brokers. Containers from arbitrary images can be created and customized via API call as well. OpenAPIv3 documentation is available on GitHub[2], along with all relevant artifacts. **Service Nodes** are atomic data reading, writing, or transformation steps. They communicate to peer nodes, and expose their internal status, through **MQTT brokers**. When created, all Service Nodes look the same: they implement the identity function, and expose APIs for their configuration. The implementation of the task that the node executes is loaded via API call from the **Transformation Libraries**, which are extendable shared repositories of reusable software components (Node-RED subflows). **Crazy Nodes** are Service Nodes that also expose APIs for arbitrary modifications to the node implementation, with immediate effect, without service disruption. The possibility of modifying the software implementation through calls made to APIs that are part of the implementation itself, reliably, with immediate effect, and without service disruption, is a unique feature of Node-RED, which makes it particularly suitable for reconfigurable manufacturing applications. **BLE Nodes** are reconfigurable Node-RED applications that turn any device equipped with a Docker engine and a BLE adapter, into a BLE peripheral. More on BLE Nodes will be said in Par. 3.3. For parallel computing, Big Data storage and visualization, Machine Learning (ML) models training and usage, Service and Crazy Nodes interface with **AI Server** nodes, relying on a specific module natively available in all Transformation Libraries. AI Servers are Scala+Spark+Akka HTTP applications. They embed extendable libraries of parallel/AI tasks. They keep a configuration record for each interfacing Service or Crazy Node, which is populated through API calls made to client nodes, and then forwarded to the server. They get input data or control signals from client nodes, and provide their outputs back to client nodes, or they publish to event streaming servers. For security purposes, all APIs make calls to external ACL APIs to validate incoming requests. Stub implementations of ACL APIs can be created through API calls to the Network Factory, as **ACL Nodes**. An example computation network that can be built, configured and operated through the above-outlined framework is depicted in Fig. 1.

[1] `https://hub.docker.com/r/msoderi/network-factory`

[2] https://github.com/mircosoderi/State-of-the-art-Artifacts-for-Big-Data-Engineering-and-Analytics-as-a-Service

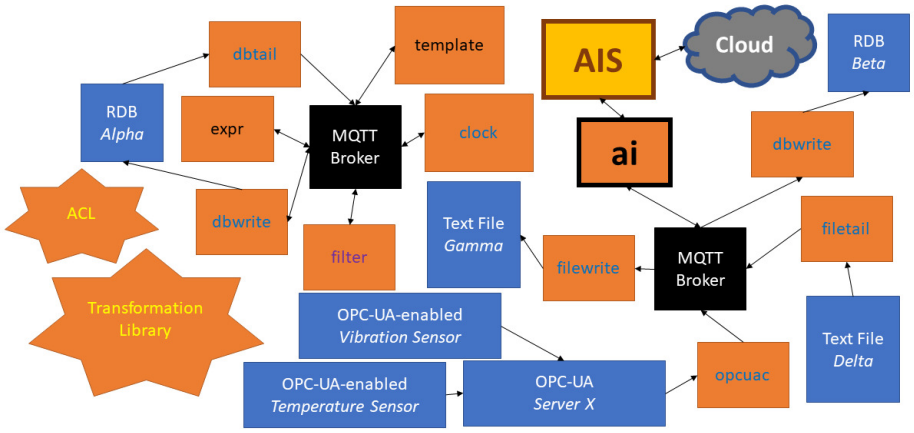

Fig. 1. An example distributed system that can be built through our proposed framework. Service/Crazy nodes are depicted each as an orange rectangle; some of them interface with IoT devices, databases, file systems... (blue labels), while others only interface with configured brokers (black labels), or with AI servers (bold large label).

3 Demo

In this section, the demo of Bluetooth Low Energy Peripherals and Edge-to-Cloud Data Assessment as a Service is presented. More specifically, the reference scenario and possible applications of this demo are identified in Par. 3.1. An overview of the involved components and their interrelations is given in Par. 3.2. The software installed on the IoT device is described in Par. 3.3. The mobile app is described in Par. 3.4. The Cloud analytics server is described in Par. 3.5.

3.1 Scenario

The reference scenario for this demo is outlined as follows: (i) multiple identical IoT devices are deployed across heterogeneous environments for some sensor-based quality assessment; (ii) both local assessment (performed separately at each device, based on its only measurements), and central assessment (based on all measurements from all devices), are relevant; (iii) then, no fixed assessment criteria can be used, and Machine Learning is needed; (iv) assessment must be performed at real-time; (v) the device moves along with the user, who uses a mobile app to display detected values and assessments, or an Android system is embedded in the device; (vi) some Internet connectivity is available; (vii) the overall application context is volatile, the evolution is not totally predictable, and requires immediate countermeasures, so the maximum possible degree of remote real-time reconfigurability is required. Example fields of application are environmental or machine monitoring, or wearable devices for biologic parameters monitoring and semi-automated therapy administration.

3.2 Overview

For demo purposes, a notebook equipped with a BLE adapter was used as IoT device, and a data input Web interface was created inside of it for emulating the generation of sensor measurements. Generated measurements go through the local assessment, before being delivered to the Android app via Bluetooth. For connecting to the IoT device, the mobile app scans a QR code, which bears the address of the BLE adapter of the IoT device. Once connected, the app starts receiving values from the IoT device, along with local assessments, and immediately starts displaying them (the text background color indicates the assessment), and submitting them to the central server for that they could be stored, and then used for training the central ML model. Meantime, in the background, via API calls, the app adds some logic to the central server, consisting in an HTTP input, Kafka clients, AI client, and MQTT client nodes. They all are client nodes; prediction model and server are shared. Once the generation is complete, the app starts sending values to the generated central logic. The app gets central assessments related to its inputted values through a dedicated MQTT broker/topic, and displays them by setting the background color of the screen (not to be confused with the background color of the text). For demo purposes, an MQTT broker instance is created, configured, and run in the central server through API calls made to the Network Factory, along with single node demo Kafka and HDFS clusters. The demo was tested with the role of the central server covered by a second notebook equipped of a Docker engine, connected to the same local network of the IoT device notebook. A high level representation of the demo is given in Fig. 2.

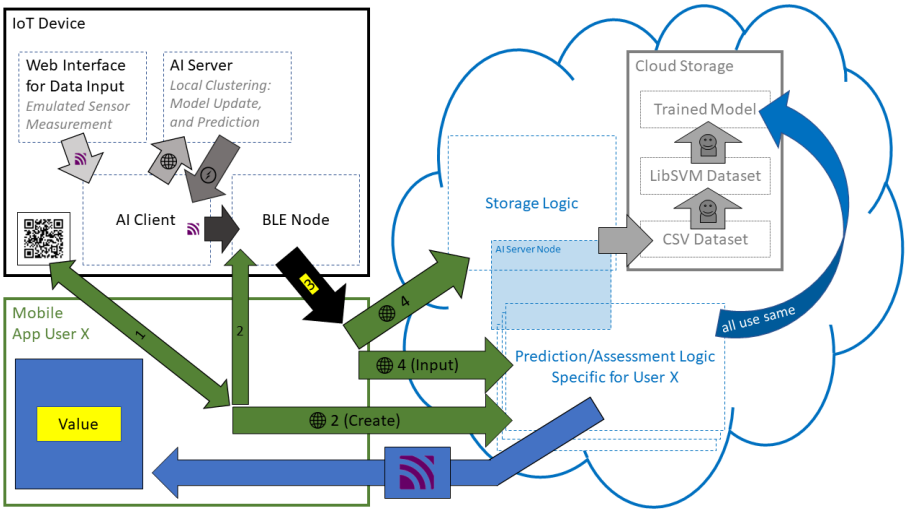

Fig. 2. A high level view of the proposed demo

3.3 IoT Device

The main takeaway from this part is that any device, even a notebook, that runs a Docker engine and is equipped of a BLE adapter, can be turned into a BLE peripheral, with all services and characteristics created (and possibly recreated) from remote. The requests in the Smart Microchip folder of the Postman collection (available on GitHub as `SmartXxxxxxxxV2.postman_collection.json`) show how that was done for this demo. What we do there, we (i) create and run a Network Factory container; (ii) create and run some utility nodes; (iii) create the data input Web page at `https://localhost:2142/ui`), to emulate sensor measurements; (iv) create the AI server, inject the task, trust the (future) client, and run; (v) create and configure the AI client, which gets input values from the local broker, interfaces with the AI server via API calls and HTTP socket, and publishes values and assessments through the local broker; (vi) create and configure the BLE node. To be noted that, unlike all other nodes, the BLE node (container) is run in host networking and with high privileges, for granting access to the BLE adapter.

3.4 Mobile App

The Android app is available on GitHub as `SmartXxxxxxxx.zip`, is the first example of automated, API-based, event-driven remote software installation, configuration, and operation, in the literature that we have produced so far in the context of our proposed framework for ubiquitous, and remotely reconfigurable data communication, engineering, visualization, and analytics. As soon as the QR code that bears the BLE device address is scanned, a new subsystem is generated, configured, and operated from scratch, from remote, via API calls, on the central system, by the mobile app. The subsystem only accepts inputs coming from the specific app installation/user by which it was generated, asks central assessments to the shared AI server (which uses a shared prediction model), and publishes them on a MQTT broker/topic dedicated to the specific app installation/user. Notably, the mobile app is a concentrate of communication technologies: (i) it receives values and local assessments from the IoT device via Bluetooth LE; (ii) it creates, configures, and operates a new subsystem in the central server via API calls; (iii) it receives central assessments via MQTT.

3.5 Cloud Analytics

The API requests in the Smart Central folder of the Postman collection generate, configure, and run the basic implementation for central assessment. Further subsystems are then added by the mobile apps via API calls at usage time, as seen in Par. 3.4. Measurements are stored in a CSV dataset. When the central administrator decides that it is time to generate or update the prediction (ML) model, they issue the requests in the HDFS Libsvm folder (which generates a LIBSVM dataset from the CSV dataset and stores that back to HDFS), and then those in the Create Model folder (to train a K-Means model using the LIBSVM dataset, and store the trained model to HDFS).

4 Conclusions

In this work, a demo was proposed where IoT devices perform a local data assessments and expose data and metadata via BLE. Mobile apps connect, forward the data to a central assessment system, and display data and all related assessments at real-time. All software is installed, configured, and operated via API requests, relying on our framework for ubiquitous and reconfigurable data communication, engineering, and analytics. Principally, the demo shows: (i) the possibility of creating BLE services via API calls, and (ii) the possibility of installing, operating, and modifying remote software, without service disruption.

References

1. Bortolini, M., Galizia, F.G., Mora, C.: Reconfigurable manufacturing systems: Literature review and research trend. Journal of manufacturing systems **49**, 93–106 (2018)
2. ElMaraghy, H.A.: Flexible and reconfigurable manufacturing systems paradigms. International journal of flexible manufacturing systems **17**(4), 261–276 (2005)
3. Koren, Y.: General rms characteristics. comparison with dedicated and flexible systems. In: Reconfigurable manufacturing systems and transformable factories, pp. 27–45. Springer (2006)
4. Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., Van Brussel, H.: Reconfigurable manufacturing systems. CIRP annals **48**(2), 527–540 (1999)
5. Koren, Y., Shpitalni, M.: Design of reconfigurable manufacturing systems. Journal of manufacturing systems **29**(4), 130–141 (2010)
6. Lasi, H., Fettke, P., Kemper, H.G., Feld, T., Hoffmann, M.: Industry 4.0. Business & information systems engineering **6**(4), 239–242 (2014)
7. Mehrabi, M.G., Ulsoy, A.G., Koren, Y.: Reconfigurable manufacturing systems: Key to future manufacturing. Journal of Intelligent manufacturing **11**(4), 403–419 (2000)
8. Morgan, J., Halton, M., Qiao, Y., Breslin, J.G.: Industry 4.0 smart reconfigurable manufacturing machines. Journal of Manufacturing Systems **59**, 481–506 (2021)
9. Soderi, M., Breslin, J.G.: Crazy nodes: Towards ultimate flexibility in ubiquitous big data stream engineering, visualisation, and analytics, in smart factories. 11th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2022) p. to appear (2022)
10. Soderi, M., Kamath, V., Breslin, J.G.: A demo of a software platform for ubiquitous big data engineering, visualization, and analytics, via reconfigurable micro-services, in smart factories. In: 2022 IEEE International Conference on Smart Computing (SMARTCOMP). pp. 1–3. IEEE (2022)
11. Soderi, M., Kamath, V., Breslin, J.G.: Toward an api-driven infinite cyber-screen for custom real-time display of big data streams. In: 2022 IEEE International Conference on Smart Computing (SMARTCOMP). pp. 153–155. IEEE (2022)
12. Soderi, M., Kamath, V., Morgan, J., Breslin, J.G.: Ubiquitous system integration as a service in smart factories. In: 2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS). pp. 261–267. IEEE (2021)
13. Soderi, M., Kamath, V., Morgan, J., Breslin, J.G.: Advanced analytics as a service in smart factories. In: 2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI). pp. 000425–000430. IEEE (2022)