Position papers of the Federated Conference on Computer Science and Information Systems, pp. 135–142
DOI: 10.15439/2017F527
ISSN 2300-5963 ACSIS, Vol. 12

CHARIOT: An IoT Middleware for the Integration of Heterogeneous Entities in a Smart Urban Factory

Cem Akpolat[1], Doruk Sahinel[1], Fikret Sivrikaya[1], Grzegorz Lehmann[2], and Sahin Albayrak[2]

[1]German-Turkish Advanced Research Center for ICT, Berlin, Germany
[2]DAI Labor, Technische Universität Berlin, Germany

***Abstract*— The main innovation behind Internet of Things (IoT) is the fact that numerous devices will be able to communicate with their surroundings and the world in general. This communication ability of devices is expected to transform the existing network infrastructure in a radical way. The massive growth of the number of connected devices with IoT and the diversity of IoT use-cases and services bring significant technical challenges to existing communication network infrastructures, as they need to integrate heterogeneous and networked devices, objects and services with different requirements. In order to overcome these issues and to realize the potential of IoT, we propose a middleware called CHARIOT, which devises a runtime environment integrating heterogeneous resource-constrained devices and sensors communicating with various protocols, and a scalable and dynamic communication layer that abstracts the connected devices and enables their intercommunication. An urban smart factory scenario is used to highlight the future IoT requirements and the need for CHARIOT.**

***Index Terms*—IoT, Device Abstraction, Directory Service, Device Heterogeneity**

I. Introduction

FOLLOWING the global interconnection of people and services through the Internet, the concept of Internet of Things (IoT) has been on the rise with the ambition to digitalize and interconnect everyday objects in many domains of life and work. The great potential of this ambition lies in the holistic networking of the involved entities, i.e., the ability of any device to interact with any other device. Services that are able to make use of device communication and the data they provide are likely to expand and offer novel opportunities in various domains such as industry, healthcare, vehicular traffic and entertainment. Nevertheless, the increasing number and variety of devices or sensors in the growing IoT world makes it ever more complex to realize and manage such holistic interconnection systems. Available IoT solutions are mostly developed either for small sized platforms or for specific scenarios with a predefined set of sensors or devices, and they are far from providing the dynamic scaling and adaptability required to realize the added value of IoT holistic networking.

Orchestrating seamless communication among heterogeneous devices is a typical challenge in the IoT domain, therefore it draws attention as an emerging research problem. In order to illustrate those challenges, we realize a *smart urban factory* testbed from Industry 4.0 (i40) domain with as many physical components as possible, without relying on heavy industry components. Smart urban factory is a future oriented factory, in which various primitive and complex devices like cyber-physical systems (CPS), software entities, robots and humans work cooperatively to produce an individual product designed by its customers. The objective of smart urban factory is to virtualize the whole factory components by abstracting all devices and systems, and ensuring an interoperable communication environment for its users. As a result of this abstraction, customers will be able to monitor their product, while remaining agnostic to underlying devices and protocols.

CHARIOT project provides a scalable IoT middleware that highlights the holistic networking of IoT entities representing heterogeneous devices or services, and demonstrates its features through a smart urban factory environment where it shows, e.g., how smart things can be connected to each other in the production line process in cooperation with humans and robots, how the warehouse stores products, and how the products are delivered. All these challenging processes include various devices ranging from primitive to complex ones, and also human actors. In this paper, we propose a new approach for the communication layer in CHARIOT project to address the interoperability in heterogeneous environment to ensure a homogeneous and reliable communication among all virtualized entities representing various devices and software with distinct characteristics.

The rest of the paper is organized as follows: Section 2 discusses the general required criteria of IoT runtime environments (REs) that connect sensors and devices with different communication protocols to each other and the Internet, and compare them based on these criteria. Section 3 gives insight into the architecture of CHARIOT middleware, followed by the novel interoperable communication approach towards i40. In Section 4 we introduce the CHARIOT smart urban factory use case, after which we share our initial results about device abstraction in the runtime environment and future work in Section 5. Finally, we conclude the paper in Section 6.

II. Background and Related Work

If an IoT platform targets a holistic interconnection system for services, devices and sensors, then it should be able to provide solutions for challenges such as heterogeneity, availability, interoperability and scalability, all of which are well-known in the IoT domain [1]. Availability of services and

devices either as software or hardware, and the trusted (non-modified) data availability at each layer of IoT must be guaranteed in such a platform. The increase in device heterogeneity necessitates not only interoperable communication among the heterogeneous devices, but also an abstraction of devices, so that the platform is agnostic to underlying devices and protocols. Life-cycle management of each service and device, including continuous monitoring, control and configuration, has to be carried out to maintain interoperability in such platforms. Finally, IoT platforms should also be designed in a scalable way, so that it is possible to add new devices and services without imposing a challenge on existing services while adapting itself to resource-constrained and resource-rich situations. For a detailed analysis of IoT platforms, the readers are encouraged to check [2].

IoT platforms are composed of many layers such as objects (device layer), object abstraction, service management, application layer and business layers [1], each of which plays a special role and performs specific tasks. The first interaction with devices occurs in the device layer, where the data are collected from attached sensors or devices and then are transmitted to the upper layers through IoT Runtime Environment (RE). IoT RE is an important component of CHARIOT, which should act as a bridge between IoT devices and the middleware. In this section, we identify available IoT REs that we believe are most suitable to CHARIOT middleware and evaluate their capabilities that should cover the following aspects: Modularity, scalability, i.e., the capability of running on a wide range of resource-constrained and non-resource constrained platforms, holistic view of the connected devices, full device abstraction and compatibility with common IoT communication protocols, unified API to access devices from IoT apps, openness, automatic app loads and updates, platform-independence, and access management to apps and devices.

Eclipse Kura [3] offers a general middleware and application container for IoT RE services. It provides a manageable and intelligent RE, on which running applications can aggregate data and share them securely with a cloud platform. IoT REs have a requirement of being able to run on any IoT device, and Kura fulfills this requirement to some extent by being able to run on various platforms such as mobile devices, desktop, wearables and Raspberry PI [2]. Kura is a Java-based platform built on top of the Open Service Gateway Initiative (OSGi) framework; therefore, it is compatible with Windows and Linux and other operating systems for which Java is available. To summarize, Kura can run on resource-constrained devices, but there is a limitation due to OSGi.

Kura offers many services for IoT devices. I/O services enable access to the underlying hardware, then the data collected from the hardware can be saved, forwarded and published at the IoT RE with data services. Kura enables direct communication with cloud platforms, and remote management feature of Kura makes it possible to manage IoT apps via MQTT protocol. Networking Service provides an API that possesses enhanced routing and networking capabilities to abstract many communication technologies such as Ethernet, Wi-Fi, and cellular networks. Last but not least, Watchdog Service monitors critical components and undertakes failure detection. The main focus of the Kura project is to bind sensors and actuators, collect data and transfer it to the cloud. It cannot satisfy CHARIOT requirements mentioned in Section III, as it is not scalable enough, the development of a device driver is not well-documented and the provided interface for devices is not user-friendly.

Alljoyn [4] is one of the emerging frameworks in IoT domain and it offers an open source framework that provides interoperability among heterogeneous devices, execution of distributed applications and dynamic composition of proximal networks. Furthermore, Alljoyn's framework targets a standard communication in mobile P2P systems. Discovering proximal devices and applications, adapting the framework based on the device characteristics, providing many connectivity technologies such as Bluetooth and Wi-Fi, and ensuring interoperability between distinct operating systems are some of the prominent features of Alljoyn. That being said, Alljoyn has some limitations to be used in a holistic IoT platform design, such as not being scalable enough to manage large scale smart IoT objects, not being able to manage big data storage and real-time analytics, and no support for the connection of devices residing under different subnets, meaning that all devices should reside under the same local network. Furthermore, a central thing manager does not exist and all nodes should connect to each other, leading to a likely increase in delay and non-reliable communication.

IoTivity [5] is an open source project aiming to enable seamless and secure device-to-device (D2D) connectivity in IoT ecosystem that is mainly supported by Samsung and Intel. The core motivation behind the IoTivity project is to bring together the open source community to speed up the creation of new framework and services required to bind the IoT devices. Its architecture provides various communication mechanisms, ensures security and identity, defines object models and APIs, guarantees interoperability between devices and applications, and connects all possible devices ranging from simple wearables up to smart cars.

The framework model of IoTivity comprises of four essential blocks: discovery, data transmission, data management and device management. Multiple discovery mechanisms for devices and resources in proximity and remotely are supported. Information exchange and control messages rely on a messaging and streaming model. Aggregation, storage and analysis of data from devices are carried out to manage the data, and device configuration, provisioning and diagnostic of devices are handled via device management module. IoTivitiy offers two different environments, one for resource-constrained and another for resource-rich devices. It runs on various operating systems such as Android, Linux, Arduino, Tizen, and Windows.

IoTivity platform is designed as connectivity-agnostic with the aid of the abstraction of connectivity layer, i.e., it supports wired and wireless connection technologies such as

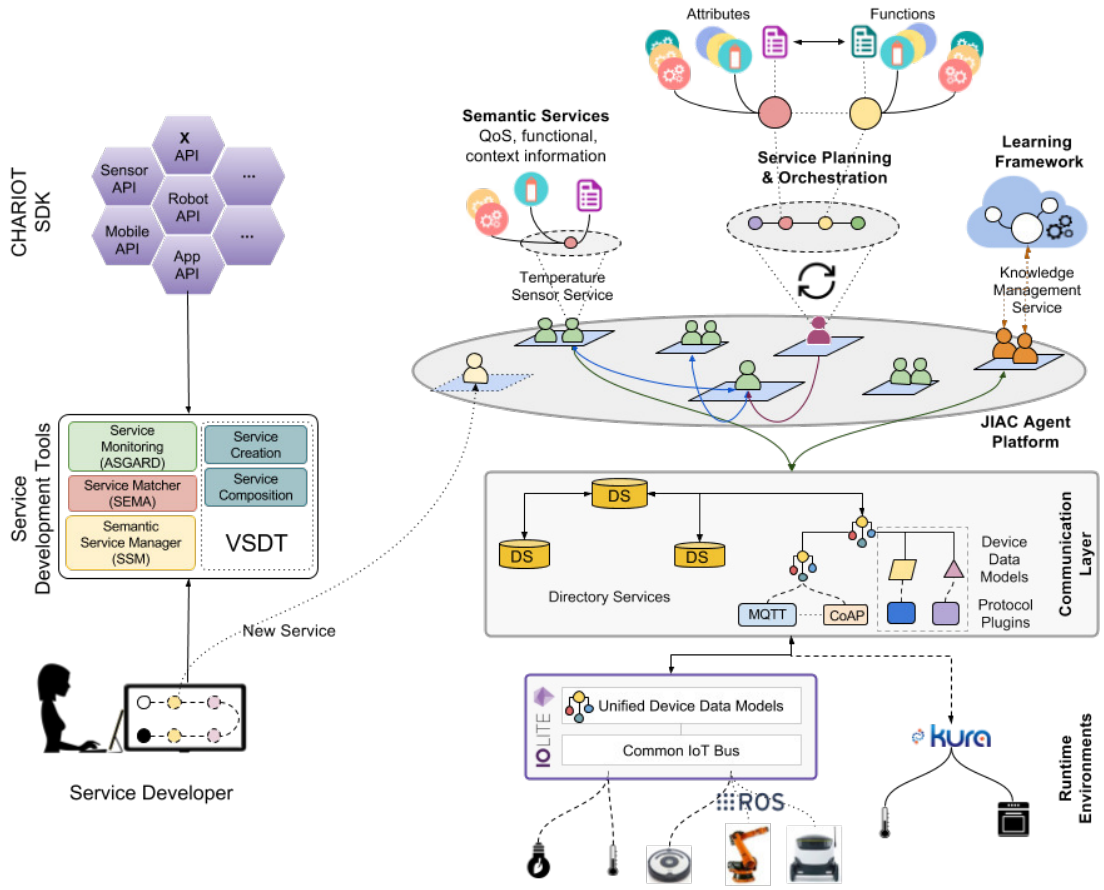

Figure 1: High-Level System Architecture of CHARIOT Middleware

Wi-Fi, Ethernet, Bluetooth-Low-Energy (BLE), Near-Field-Communication (NFC). Connectivity-agnostic layer of IoTivitiy makes it favorable in comparison to other platforms; however, it still does not conform to the desired CHARIOT middleware needs, as it only focuses on D2D communication and does not consider how an added value can be created from these interconnected devices via novel applications.

IOLITE is a Smart Home - Smart Building platform, providing the foundation for an open smart home ecosystem [6]. It offers the possibility to integrate devices of many kinds and to use them in a variety of applications in a wide variety of ways. IOLITE itself is a closed source environment but offers several APIs for extending the platform (i.e., an interface for developing apps to control the system as well as extension points for supplying new drivers for devices and sensors, if not already provided). The main features that characterize the IOLITE Platform are open SDKs for extension, offline capability, support for different user types, smartness, adaptivity, extensibility with new apps and drivers, and a customizable and modular structure.

Even though IOLITE is orginally designed for smart home environment, its flexible architecture enables the integration of new custom devices, thus its usage area can be expanded to other IoT domains. Furthermore, IOLITE abstracts all heterogeneous devices and represents them with a common device model in its ecosystem. Through this abstracted device model, CHARIOT middleware does not have to deal with the issues stemming from device heterogeneity. IOLITE RE differs from the other analyzed REs by providing a user-friendly platform, user management for devices that can enable access rights management, and an app-store that enables deploying many applications that can make use of the properties of the connected devices. Moreover, IOLITE provides a unified programming interface that makes the implementation of a driver and an application quite simple. Last but not least, the automatic update feature of IOLITE is seamlessly operated without touching any configuration file, once the new version is available.

III. CHARIOT Architecture

CHARIOT offers an IoT middleware that encompasses many layers ranging from device layer to service layer. Device layer behaves as an IoT gateway for various devices that communicate with different protocols, and abstracts device features for other layers. The communication layer provides reliable communication for the services that represent devices, and enables device querying through their semantic

service descriptions. These descriptions are recognized by the semantic service layer of CHARIOT. The management and orchestration of semantic services is carried out by the service planning and orchestration component. This component offers context- and location-aware and error-resistant planning for operations by continuously monitoring QoS parameters of services and by reacting to the dynamic changes that occur, for instance, in case of a service-chain crash. This highly dynamic environment requires autonomous and adaptive behaviors for services, which is provided in CHARIOT by the knowledge management layer. Knowledge management layer adds learning capability to services and makes it possible to transfer the obtained knowledge among the services. The protection of those services and their access to the devices are carried out by the security layer that manages the access control mechanism.

To improve the ease of use for an IoT platform, an extendible platform is required in addition to the above mentioned constructive layers. CHARIOT middleware provides an SDK that facilitates the integration of any device ranging from primitive to sophisticated device, software entities to human actors, and a number of software development tools that deal with the monitoring, maintenance and control of those services. All these layers and tools are designed on top of an agent platform that forms a distributed service structure. The architecture of CHARIOT, which covers all layers and tools, is presented in Figure 1.

As mentioned above, one of the essential goals of CHARIOT is to highlight the holistic networking of IoT entities representing devices or services on the platform with device-heterogeneity support and ensure their continuous availability. In this study, we focus on the RE and communication layer of CHARIOT middleware, where device heterogeneity and scalability are ensured.

A. Runtime Environment

CHARIOT uses IOLITE as an RE to interact with IoT devices, as mentioned in Section 2. In this section, we explain how device integration into CHARIOT middleware is achieved by using IOLITE. The integration of devices over IOLITE in CHARIOT middleware requires two steps from the device layer perspective. First, the device drivers implemented for IOLITE are integrated and then the integration of IOLITE RE to CHARIOT middleware via Message Bus Protocol is carried out as described below:

1) Device Driver Development and Integration: The realization of smart factory use case scenario requires the development of a number of special drivers for sensors and actuators. During the driver development phase, first the feature of the devices and the supported communication protocols are identified. As the communication protocol varies from device to device, i) RE should be capable of extending itself to support different communication protocols such as Wi-Fi, BLE, Zigbee. ii) The data transmission between device and RE has to be solved by supporting different data transmission protocols like CoAP, MQTT, REST, SNMP, ModBus, TCP, UDP. iii) Through the established communication and data transmission in XML, JSON or another format between RE and device, device functionalities can be mapped to RE device model concept, thus abstracting it for the upper layers.

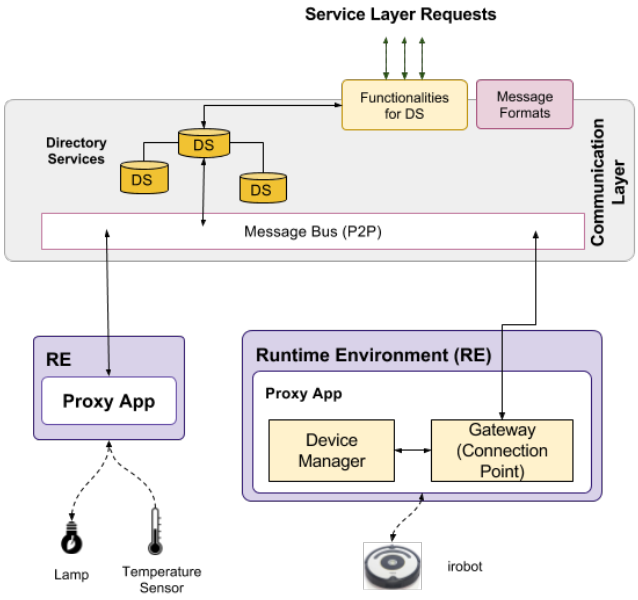

Figure 2: Integration of RE to Chariot Communication Layer

2) Integration to CHARIOT Middleware: The future-oriented IoT middleware needs a system in which IoT devices represented by software entities should interact with each other to create new value-added services and apps. The creation of this interaction environment necessitates first the integration between REs and CHARIOT middleware to map the physical devices to their virtual entities. All these communication processes and seamless integration would be performed through Message Bus. To build a bridge between RE and CHARIOT middleware over the Message Bus, a proxy RE application, depicted in Figure 2, is required in order to initiate the communication to CHARIOT communication layer and later to transmit devices' data and their models. This initiation message should include RE identifier, its device identifier and device property identifiers. In case the value of device property is accessed, these three identifiers have to be provided to access the device with another message type, which will be defined in the system architecture.

The massive data generation from devices, their analysis and the extraction of valuable information to optimize the processes in the smart factory may overstretch the limits of CHARIOT cloud and it might be possible that the minimum latency required in certain smart factory scenarios cannot be guaranteed. To address this issue, the proxy RE app will transfer some service functionalities using fog computing approach [7] to the device layer, i.e., to the RE that serves as an IoT gateway, to decrease the latency and enable data processing at the edge without transmitting the data to the cloud. With the help of this approach, IoT RE itself would be responsible to

decide for the amount of data to be stored from the devices, the frequency of data updates, registration and removal of devices, etc. In case communication between two IoT-REs is needed, it can be established via P2P communication.

B. Ensuring Interoperability in Communication Layer

Communication layer in CHARIOT matches incoming service requests with relevant IoT entities, and it promises scalable communication between different REs to enable P2P links between IoT devices. Three different software modules are provided to CHARIOT developers in the communication layer, so that they can interconnect all high-layer services to low-layer devices and entities regardless of what communication protocols they use and the environment in which an application is developed.

1) Message Format: The first software library involves application independent formats for messages between the RE that the devices or their software units are registered and the directory service (DS). This messaging format is used for device registration, removal and status updates.
2) Scalable Directory Service: DS is responsible for providing information about devices that are connected to the CHARIOT middleware and for storing the relevant path that enables the communication with these devices and accessing their current status. Device information is stored as OWL-S descriptions [8].
3) P2P Communication: Another essential feature of the communication layer is establishing communication channel between entities connected to CHARIOT. For this reason, a software library is formed to cover P2P messaging and security related functionalities.

The communication components given above are pictorially represented in Figure 2. In the following subsections, we describe these components in more detail.

1) Message Protocol Definition: Messaging formats in the communication layer are specifically defined for the interactions between the RE and the DS. In CHARIOT architecture, the RE collects all the device information and then forwards this information to DS by using this messaging protocol. This messaging protocol should also be able to inform the DS about errors and unknown messages. The modules that are used between the RE and the DS for messaging is shown in Figure 3. A communication protocol library is constructed in CHARIOT to be used by a protocol adapter that provides interfaces for different messaging protocols such as MQTT and CoAP. The RE can choose an interface defined in this adapter to communicate with the DS. A new interface must be added to this library, should an RE choose to communicate via a new protocol. The content of the messages that reach the communication layer should be modeled by the RE beforehand, so that the message broker and analyzer in the communication layer can extract the data in these messages and forward them to the related domain within the DS.

2) Distributed and Scalable Directory Service: DS stores device information in a location and content identifier for information-centric networking (ICN), so that other

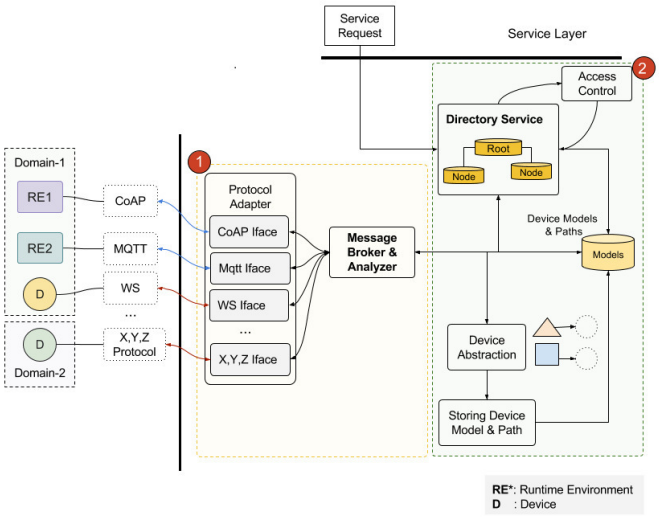

Figure 3: Communication Layer Architecture

devices in the device layer and the services that want to make use of the device can locate this device within the CHARIOT framework. The device description in the DS is taken from IOLITE Property Profile Model and converted to a URL-like address (e.g., *Icn://de.gt-arc.iot/sensor?lat=35,lon=11,radius=1km,scale=census*). This device information is also used during end-to-end messaging between devices and their software agents. As all devices in CHARIOT are represented as software agents in the service layer and are defined with semantic OWL-S descriptions, semantic search can be used for finding devices and enabling end-to-end messaging. For this reason, DS has to enable messaging both in the device layer and in the service layer. As explained in the previous subsection, device layer messaging happens between the RE and the DS, and the content of the message includes device information stored in the DS. In addition, an agent in the service layer can query the DS to establish P2P communication with another device. Service layer to DS communication involves semantic search queries from the service layer to find the relevant device information. In our DS, Semantic Service Matcher (SeMa) [9] is responsible for finding structural, logical and semantic relation among services and returning search results for these queries.

A distributed DS design is considered in CHARIOT in order to meet the scalability and latency requirements and to provide an increased performance for the platform. The design consists of DS nodes that are distributed among different domains, such as a common property (e.g., energy consumption) or a common location (e.g., warehouse). All these DS nodes are connected in a hierarchical manner to a root DS that keeps track of the devices within the CHARIOT framework and is responsible for DS monitoring, synchronization and update tasks.

Root DS should be deployed as a cloud-based scalable entity

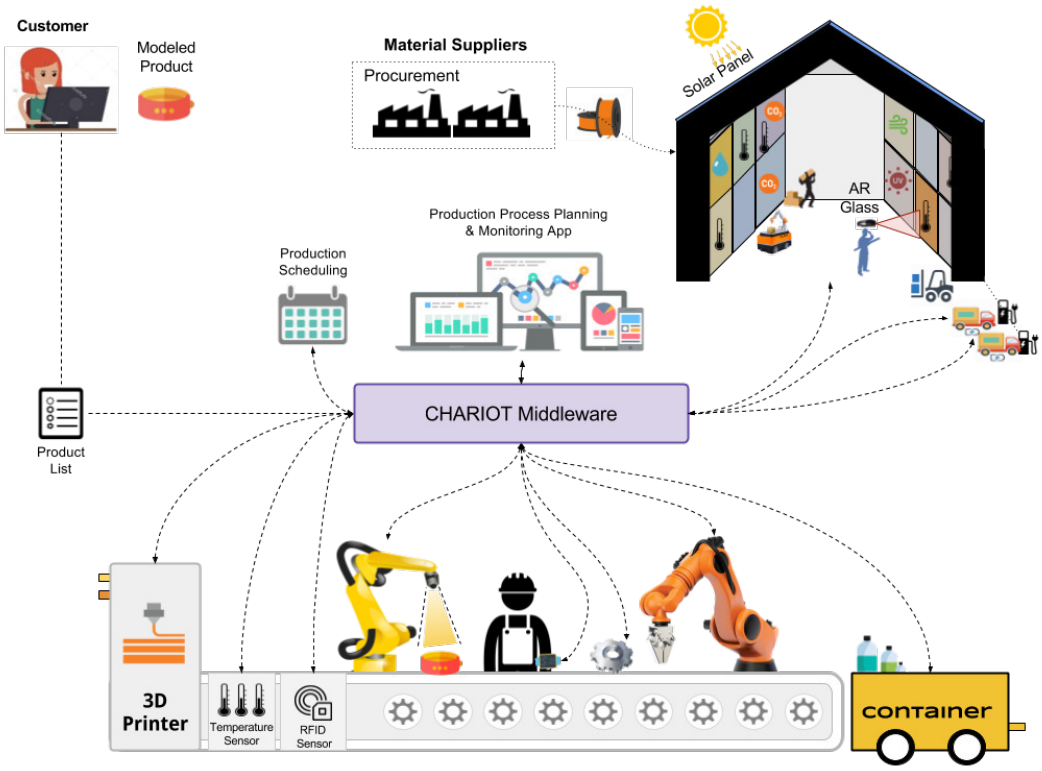

Figure 4: Industry 4.0 Urban Smart Factory Use Case

and should be able to distribute the load on a DS when required. In addition, data caching and sharing functions will be designed for the DS to increase the efficiency and to overcome response time constraint [10]. With all these functionalities, DS forms the backbone of the decentralized, hierarchical and scalable communication infrastructure of CHARIOT, which extends across multiple administrative domains.

An ICN-based architecture is foreseen for the DS design in CHARIOT. In such an architecture, DS nodes serve as ICN routers, where data packets are routed based on their contents in order to match service requests to suitable IoT devices or vice versa [11]. DS nodes can also cache information to speed up querying and data transmission in general. The main advantage of an ICN-based DS structure is its built-in content-based search and caching functions, which is very helpful for service discovery from the service layer. Device registration can either occur at the closest directory node or at the root directory node and then the root directory chooses the most suitable storage node for the device based on the device description.

3) P2P Communication: P2P communication can take place as the underlying data exchange for agent-to-agent communication in the service layer, or as direct communication between devices that require data from other devices at the device layer. In the first case, two agents communicate via ActiveMQ protocol at the service layer after searching for the matching service description in the DS. In the second case, the communication layer is only responsible for data exchange between two REs, as the data passes through the Message Bus. Once the message reaches the RE, communication from the RE to the device is handled internally by the RE. The essential role of this component is to ensure a secured communication channel (Message Bus) between RE and CHARIOT middleware with SSL/TLS or another secure method, while DS is in charge of providing registered device addresses to REs. In addition, the interaction between IoT devices running on different REs (RE-to-RE communication) via P2P communication in a common message format should be defined to provide seamless interaction between devices.

IV. Smart Urban Factory Use Case for CHARIOT

Industry 4.0 introduces a new manufacturing perspective that uses new technologies and devices that can autonomously communicate and exchange data among each other. In a smart urban factory model, where the decisions are taken in a decentralized fashion thanks to autonomous systems, virtualized copies of physical devices and processes are monitored by more complicated computers, fulfilling almost all requirements of i40. As i40 has a direct relation to Cyber-Physical Systems (CPS), IoT and cloud computing, its improvement is also indispensable, even though it is in the early stages of its development. Moreover, increasing device heterogeneity and their varying requirements (e.g., CPU, bandwidth, memory),

the enhancement in analytic functionalities that enable better estimation on the device layer, and generation of efficient business applications with human-machine integration would foster the developments in i40. In the following, we introduce a use case scenario that brings mass customization, 3D printing, predictive maintenance, human-robot interaction, warehousing and augmented reality features together to demonstrate a smart urban factory environment and show how our approach aids in solving its challenges.

In a smart urban factory, we highlight the mass customization [12] feature of i40 and how other i40 features are harmonized with each other. The main idea of the mass customization is to construct a smart factory, as illustrated in Figure 4, which allows customers to create individualized products in any material and form. Customers may have the opportunity to produce a variety of products, ranging from bracelets to certain spare parts required for a machine. Customers use an online drawing platform that enables 3D modeling of the product design, which is then uploaded to CHARIOT middleware. CHARIOT schedules the printing time of the product models by taking many parameters such as priority, 3D printer and cartridge material availability, and energy consumption into account. These parameters' data are retrieved from all devices in the smart factory through REs and then unified in CHARIOT communication layer. The scheduled product models with a unique identifier are printed in 3D printers and on a passive RFID chip containing ID and additional production information. 3D printer receives the printing order from printing service, which can directly access the 3D printers and their functionalities over RE. Once this process is completed, RFID reader helps direct the product to the corresponding conveyor belt. Before boxing the product, an inspection process through the camera should be performed to detect whether the product meets the requirements of the customer design. This process requires object recognition and image processing operations, for which the inspection camera establishes a communication over its RE and communication layer with a service in CHARIOT middleware that enables data processing either in the local or in the cloud. After a successful inspection, the product is taken either by a robot or human worker from the conveyor belt either to be delivered to the delivery truck or to be stored in the warehouse. In the warehouse, the products are stored and preserved with respect to their characteristics by using different sensor technologies such as temperature and humidity control. A continuous data aggregation through warehouse's REs and CHARIOT communication layer is performed as well, and then all data are processed in the CHARIOT middleware to react to any change or an abnormal behavior of the warehouse's sensors or actuators, such as sensor replacement or repair.

V. Initial Results & Future Work

CHARIOT communication layer has two main tasks, namely, device abstraction and enabling communication between all devices connected to CHARIOT middleware. The first phase of the project focuses on the device abstraction and in this section we present our initial work conducted in this phase thus far.

Device abstraction phase involves two steps: *i)* modeling a device and its features using *Profile Property Identifier (PPI)* and matching the device functions to the PPI, *ii)* transmitting the abstracted device data via proxy application to the communication layer. For the initial task, we modeled and implemented many devices such as charging station, solar panel, gate, parking sensor, motion sensor, smart meter using IOLITE SDK and PPI. We are now able to abstract devices using *IOLITE Device API* that returns a JSON device data model along with its current values. The implemented devices are depicted in Figure 5. In the second step, we design an IOLITE application that can transmit the device data model to the requester and receive requests from the communication layer or from other devices to execute the functions of connected devices using P2P communication.

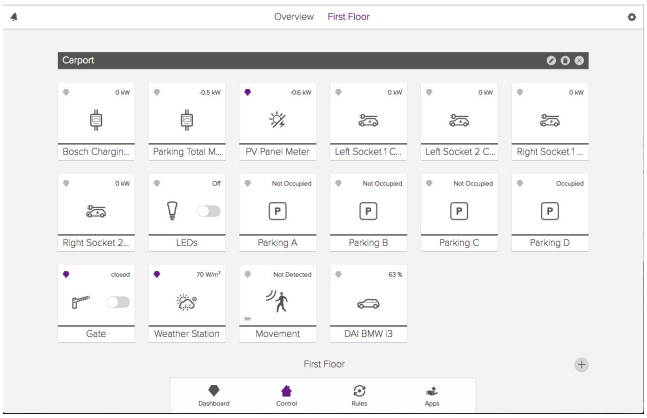

Figure 5: IOLITE Device Control Panel and Representation of Devices

This work mainly focuses on the CHARIOT communication layer; however, there are also other important cornerstones that play a crucial role in realizing the IoT middleware. For instance, the representation of devices as a service on JIAC agent-platform [13], the enrichment of services with semantic descriptions, the planning and orchestration of services and a learning engine that provides learning capability and knowledge sharing to services are among the ongoing and planned activities of the CHARIOT project. Furthermore, the project will provide software developers a CHARIOT SDK and a set of service development tools, which facilitate the creation, maintenance and monitoring of services.

VI. Conclusion

The emerging IoT devices and their increasing variety necessitate an interoperable communication layer in IoT domains. In CHARIOT project, we reflect the device heterogeneity through a smart urban factory use case. To strengthen our approach, we analyzed available IoT REs that are responsible for device connections, and justified the selection of IOLITE as RE. Then, the identified requirements of Industry 4.0

regarding smart urban factory use case are studied. To meet the identified requirements, we proposed CHARIOT device-agnostic communication layer, in which heterogeneous devices are abstracted in cooperation with the device layer. We shared our initial results, where we abstract devices through IOLITE RE, and we finally discussed our planned activities.

ACKNOWLEDGMENT

The work of authors at GT-ARC is supported in part by the German Federal Ministry of Education and Research (BMBF) under the grant number 01IS16045.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 10 2015, doi:10.1109/COMST.2015.2444095.

[2] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, Feb 2016, doi:10.1109/JIOT.2015.2498900.

[3] "Eclipse Kura," http://eclipse.github.io/kura/doc/intro.html (accessed: 2017-05-29).

[4] M. Villari, A. Celesti, M. Fazio, and A. Puliafito, "Alljoyn lambda: An architecture for the management of smart environments in iot," in *2014 International Conference on Smart Computing Workshops*, Nov 2014, pp. 9–14, doi:10.1109/SMARTCOMP-W.2014.7046676.

[5] J. C. Lee, J. H. Jeon, and S. H. Kim, "Design and implementation of healthcare resource model on iotivity platform," in *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct 2016, pp. 887–891, doi:10.1109/ICTC.2016.7763322.

[6] "IOLITE," www.iolite.de (accessed: 2017-05-29).

[7] M. S. D. Brito, S. Hoque, R. Steinke, and A. Willner, "Towards programmable fog nodes in smart factories," in *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, Sept 2016, pp. 236–241, doi:10.1109/FAS-W.2016.57.

[8] "OWL-S: Semantic Markup for Web Services," https://www.w3.org/Submission/OWL-S/ (accessed: 2017-06-19).

[9] J. Fahndrich, T. Küster, and N. Masuch, "Semantic service management and orchestration for adaptive and evolving processes," *International Journal on Advances in Internet Technology*, vol. 9, no. 3&4, pp. 75–88, 2016.

[10] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "Dns performance and the effectiveness of caching," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 589–603, Oct 2002, doi:10.1109/TNET.2002.803905.

[11] I. Abdullahi, S. Arif, and S. Hassan, "Survey on caching approaches in information centric networking," *Journal of Network and Computer Applications*, vol. 56, pp. 48 – 59, 2015, doi:10.1016/j.jnca.2015.06.011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804515001381

[12] F. S. Fogliatto, G. J. da Silveira, and D. Borenstein, "The mass customization decade: An updated review of the literature," *International Journal of Production Economics*, vol. 138, no. 1, pp. 14 – 25, 2012, doi:10.1016/j.ijpe.2012.03.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925527312000989

[13] "Java-based Intelligent Agent Componentware," http://www.jiac.de/agent-frameworks/jiac-v/ (accessed: 2017-05-29).