A multi-domain networking infrastructure enabling the situation-specific choreography of decentralized things

Marlene Gebhard, Michael Schluse, Martin Hoppen, and Jürgen Roßmann

Institute for Man-Machine Interaction, RWTH Aachen University,
Ahornstraße 55, 52074 Aachen, Germany,
gebhard@mmi.rwth-aachen.de,
www.mmi.rwth-aachen.de

Abstract. The Internet of Things (IoT) is a network between physical and virtual things, software services and people. The specific services these things offer to set up value-added chains are provided by the computer resources they use – like cloud computing for computationally intensive tasks or edge computing for real-time processing. Concepts such as the Digital Twin enable things to operate autonomously and connect independently. This leads to self-initiated systems which require the development of new infrastructure concepts to combine different computer resources and various Digital Twins. This paper proposes the Smart Systems Service Infrastructure (S³I) – a multi-domain networking infrastructure which allows things to choreograph themselves situation-specifically in highly decentralized and dynamic environments.

Keywords: IoT Infrastructure, Digital Twin, Choreography, Value-Added Chains

1 Introduction

In the last decade the term Digital Twin emerged and several definitions were proposed. The glossary of platform Industry 4.0 defines Digital Twins as virtual digital representatives of physical assets [6]. More detailed perspectives focus on simulation or the networking of physical assets [2, 5, 14]. In [11] the definition is extended to a representation which includes semantics and is used as a structural element. Depending on the domain-specific development, the scope of operation offered by the Digital Twin varies. In addition to that, the question how Digital Twins can be used to set up decentralized value-added chains, where Digital Twins live, how they interact with software services and humans in a holistic approach is still being researched intensively.

In this work the Digital Twin is considered to represent one asset in the IoT. Figure 1 shows the individual components of the IoT in an hierarchical order. At the lowest level are physical assets that are connected to their Digital Twins via sensor/actuator networks. The corresponding Digital Twins as well as software

Fig. 1. The Digital Twin in the IoT. Figure based on [10].

services and Man-Machine Systems (MMS) are shown in the IoT-levels above. These *things* – the *Digital Twins* representing assets, the *software services* and the *MMS* – are the nodes of the IoT. MMS provide Human-Machine Interfaces (HMI) which enable the operators to interact with the IoT-nodes in the specific field of application.

For this, Digital Twins, software services and MMS offer various services according to their specific roles as nodes of the IoT. Applications are configured and tasks are carried out because the individual participants of the IoT network and collaborate with each other creating problem-specific configurations of IoT-nodes making up the “applications”. It is the purpose of all the things – and especially the Digital Twin – to connect with other things i.e. Digital Twins, software services and MMS to create such applications and perform certain tasks. This way, concrete applications are *configured* by selecting and networking the things provided by the IoT.

All things – Digital Twins, software services and MMS – have different requirements on the computer resources they use. For a machine which has to perceive and react to its environment with the help of a sensor network, a reaction in real-time is only possible if the processing takes place on the machine itself. However, space and energy resources might be limited. Therefore resource-intensive calculations that do not have real-time requirements are outsourced to calculation services which are reached via the IoT. This illustrates that only the combination of different computing resources enables the collective execution of a task.

The sensor-example above describes one thing's interaction with one software service. The example can be expanded to a whole network of things where a multitude of Digital Twins, software services and MMS connect with each other, thus creating applications solving different tasks. Individual things provide their functionality to diverse applications, resulting in a dynamic network which adapts situation-specifically to particular tasks. Orchestration in which the process logic is implemented by a sequence control that calls up the individual processes is no longer possible. Things act autonomously and pursue various goals independently of each other. The *orchestration becomes a choreography* in which the individual things contain the process logic.

Summing it up, an infrastructure is needed that allows the things to choreograph themselves using the computer resources they need. The work at hand presents such an infrastructure which we will call S³I in the following. For that purpose, in the following section, the state of the art is pictured. In Section 3 the requirements for and the modules of the infrastructure are presented. An example application is given in Section 4. We conclude the paper in Section 5.

2 State of the art

There is a wide range of publications on the individual computing concepts (e.g. cloud computing and edge computing) [1, 8, 13, 15]. Approaches that combine the various computer resources, ranging from the cloud to edge devices, in one single infrastructure are rare, though.

Alam et al. [2] propose a combination of cloud and edge computing. They use the cloud for the Digital Twins to ensure the scalability of storage, computation, and cross domain communication capabilities. The physical layer which represents the edge of the network provides real-time sensory fusion. Two things can establish peer-to-peer (P2P) connections either through direct physical communications or through indirect cloud-based Digital Twin connections. The combination of a fuzzy rule base with a Bayesian belief network forms a decision algorithm that selects between the physical-physical interaction, the cyber-cyber interaction and a hybrid way which uses both, the physical and the cloud-based communication.

In addition to scientific approaches there are industrial approaches. The Bosch IoT Things Platform suggests an IoT Gateway Software which provides autonomy and intelligence at the edge [4]. The gateway software supports current connectivity protocols so that the IoT devices can connect to the gateway and benefit from its computing resource. On the other side the gateway offers a local abstraction and allows unified access to the connected devices and their data. Cloud Computing is used as a back end to handle tasks with advanced computing requirements.

A combination of cloud, fog and edge computing is done by Bierzynski et al. [3]. Using improved IoT hardware, edge and fog things can handle not only functional tasks but data processing and analyzing tasks like filtering and determining correlations as well. Therefore, the aim is to integrate the edge and fog things and to distribute the workload optimally on all levels (cloud, fog, edge).

For the distribution of tasks machine learning is proposed. As an example the light control of a fair hall is used. Within that example things are orchestrated by a central unit with no need for self-choreography. This approach does not answer the question how the infrastructure looks like supporting heterogeneous things which act autonomously and choreograph themselves.

All in all, from the authors point of view, concepts to combine different computing platforms exit in principle but there are no infrastructural concepts which allow autonomous things distributed over the different areas of the IoT (cloud, edge, etc.) to choreograph themselves.

3 Smart Systems Service Infrastructure (S³I)

To allow highly decentralized things which use various computing resources to choreograph themselves there are several requirements for the infrastructure these things are using.

1. The infrastructure shall allow the things to find each other. By finding each other the things can identify the participants within the infrastructure enabling the first step to choreograph themselves.
2. The infrastructure shall allow the individual things to interact. The aim is that the things first find each other and then communicate with each other in such a way as to enable them to choreograph themselves.
3. The infrastructure shall support a dynamic configuration of things. This requirement allows the things to act flexible and to react dynamically to changes within the network.
4. The infrastructure shall provide security for the participants and their data. This includes protection against malicious attacks by third parties as well as confidentiality and integrity of data.
5. The infrastructure should offer a means of communication. Therefore the participants do not have to implement their own communication infrastructure and might use the provided one instead.
6. The infrastructure should have low overhead and should not demand extensive computing resources. This way, the infrastructure can be put together using edge devices only.

The S³I answers these requirements with different modules (Figure 2), which are explained in more detail below.

3.1 Directory

The directory is the infrastructure's core. It provides a central metadata collection for the network's things, which can be searched for, created, modified and deleted according to the client-server principle.

To ensure confidentiality of this data the directory has to provide authentication and authorization of the thing or human-being (called *user* in the following)

Fig. 2. Structure of the Smart Systems Service Infrastructure. IdP = Identity Provider.

accessing the directory. The authorization part uses password or token authentication to assure that the user is the one he claims to be. Once this is verified, the directory gives access to users on the basis of their respective roles. Access can be restricted or granted down to individual directory entries and is divided into read and write permissions.

In addition to the confidentiality requirement, the directory also enables interaction between the things. Therefore, the directory does not only collect the things data but also provides their communication interfaces via so-called *endpoints*. The endpoints represent the interfaces of the things through which they make data and services available for other things. To be communication protocol-agnostic, the directory offers all data which is needed to access these interfaces. For example, this additional data might include a topic which is used within the MQTT protocol.

Apart from the protocol-specific information on the endpoints, information on the data behind the endpoint has to be available. This information indicates the structure of the data and the format in which it is presented and using the data model introduced in the following. Additionally the directory enables the endpoints to administrate data structure and data format themselves. That way, dynamically changing data models within the things as well as dynamically changing thing configurations can be accommodated for. This also allows the introduction of undisclosed things.

Data model Figure 3 shows the conceptual data model for the S³I directory. The root represents the entry point to the data and comprises the managed entities and things. Entities are users or groups of users that own, manage or otherwise relate to one or more things of the IoT. For the sake of clarity, user and group details are omitted. Things can be associated with a location to allow for spatial queries (e.g. find all nearby assets in a certain radius). Furthermore, each thing can be associated with communication endpoints for access through connectivity protocols like OPC UA, MQTT, HTTP or S³I broker. Things might also be hierarchically structured with other (child) things that are separately managed within the S³I directory. For example, a machine thing might consist of several machine child things like an engine or a chassis. Finally, the structure of a

Fig. 3. Data model of the S³I directory.

thing can be disclosed in more detail within the directory. For that purpose, the utilized data model can be specified. In more detail, (parts of) the thing structure can explicitly be exposed using a generic object model built from objects, values and links between objects. This allows to associate endpoints directly to the object or value they provide access to. This simple object model serves as a meta model on which most data models can be mapped thus supporting arbitrary application domains in the S³I directory. Providing only meta data results in low overhead and reduced computational requirements, so that the infrastructure can be set up using low-power devices only.

Implementation From a technical point of view, the directory is a database that provides an API and takes care of authorization. This set-up can be provided by open-source tools like Eclipse Ditto. Eclipse Ditto is an open-source framework for Digital Twins in the IoT. Ditto focuses on solving the responsibilities a typical “back end” has in IoT solutions [7].

To store the data, Eclipse Ditto uses a MongoDB. Besides the data of the directory entries, Ditto also stores user information to authenticate and authorize

the user. The modeling of the users as entities is shown in Figure 3 and mentioned above. The authentication of the users is done by user credentials or JSON Web Token (JWT) which correspond to a digital ID card. Authorization takes place via so-called policies, which are assigned to the corresponding directory entries of the things. A policy lists the specific roles and defines the resources the respective roles can access. The resources can be whole entries of the directory or parts within the respective entry. For every resource read or write permission can be granted or revoked. Using a JWT, Ditto authenticates the user and separates its role (e.g. owner or administrator) from the JWT. In combination with the policies, the access rights to a requested entry are described.

Ditto provides a REST API which offers the possibility to configure the policies as well as the specific entries. The computing resources on which the directory is based must therefore have IP-capable interfaces. Besides that, the directory can be set up anywhere, no matter if it is a cloud or an edge device.

3.2 Identity Provider

While authorization in the directory is done using these policies, for the authentication the user (i.e. a thing or an human-being) has to specify his data with user credentials or a token.

Using user credentials, thus password authentication, the directory would have to store the user credentials of each user. On the other side the user has to have the corresponding password for the directory ready each time another thing is invoked or some other call is made. In a network of services and Digital Twins where a lot of access points are used this is not convenient. Also, the user data, as very confidential information, would be sent along every time a request is sent.

For these reasons the S³I provides a central Identity Provider (IdP). The IdP manages the user credentials like passwords and certificates. Each user has an identity within the IdP, which is valid for other services at the same time. The identity is carried in form of a digital ID card, which is implemented as a JWT, as explained above. This digital ID card is the access key for the various things within the S³I and authenticates the user.

The login of the respective users is done with OpenID connect, which provides an authentication layer on top of the OAuth 2.0 authorization framework. OpenID connect completes the OAuth 2.0 authorization flow with authentication mechanisms [9].

3.3 S³I Broker

With the Directory and the Identity Provider participants are given the ability to make their interfaces securely accessible to others. The actual data transmission is handled by the participants themselves. To provide one message-based communication option the S³I offers an Advanced Message Queuing Protocol (AMQP) broker.

AMQP supports a request/response and publish/subscribe architecture. In cooperation with the messaging broker the protocol ensures a robust data transfer

and allows asynchronous communication by enabling messages to be stored in queues. Within the broker an exchange receives the messages and routes the data into the correct queue. The queue, in which the message belongs, is determined by a binding. The exchange also offers broadcasting of messages and topic-specific forwarding. Although the exchange types and the bindings offer a variety of communication architectures there can be custom exchange types like geodata-base routing.

4 Application

One field of application is the forestry and timber industry. It offers a multitude of different stakeholders and a heterogeneous structure [12]. In addition to forest rangers, forest workers and forest owners, facilities such as sawmills and forest machine manufacturers operate in the cluster. Regarding machinery there are forestry machines like harvesters, forwarders and skidders as well as hand-operated machines like motor saws and brushcutters. Furthermore there are services for forest growth simulations and tree species classification.

The machines might have real-time applications running on the machines themselves thus representing edge computing. Services like forest growth simulation might run in the cloud using the computing power provided here.

These things build a highly decentralized network with heterogeneous computing resources. Still they act independently and choreograph themselves to perform solutions in constant exchange.

One example to illustrate the core concepts of the S³I is the communication between a sawmill and a harvester. The sawmill determines its need for a specific amount and type of wood. The aim of the sawmill is to place this order with a harvester. Optimally, an order to a harvester which is currently in a suitable forest area and can process the order is created. The sawmill logs into the directory and queries the harvesters within a specific area using a location based search. The directory entry which is shown in listing 1 provides information on the endpoints offering the current work status of the respective harvester. After choosing a suitable harvester the sawmill queries the directory entry of the harvester and uses the denoted endpoint to place its work order.

This example illustrates how the directory is used. Using the REST API, things can query the directory and search explicitly or categorized for things. Each thing can indicate its interfaces declaring endpoints in the directory which is reduced to an AMQP endpoint in the example. Giving information about the exchange and queue which is used in the example, a communication partner is provided with the necessary information about the interface.

With the login at the Identity Provider, the user gets an ID token to authenticate at the Directory, which allows access to the information about the AMQP endpoint of the harvester. This assumes that the harvester has listed the sawmill in the directory as an authorized party before. Since the harvester controls its policies (e.g. using the API of the Directory), it determines the users of its interfaces by itself.


```

1  {   "name": "HarvesterJD_OPS",
2      "identifier": "tzccokgc-resc-glpodnchfgtq",
3      "location": {
4          "lat": 51.454602,
5          "long": 8.006526
6      },
7      "objects": [
8          {   "class": "workOrder",
9              "identifier": "tzccokgc-resc-rpkmxnwurzgd",
10             "endpoints": [
11                 {   "type": "sssiBroker",
12                     "uri": "amqp://sssi/forestAuthority",
13                     "queue": "harvesterJD_OPS",
14                     "exchange": "sssi"
15                 }
16             ]
17         }
18     ]
19 }

```

Listing 1: Directory entry of the harvester.

5 Conclusion

This paper presents an infrastructure for highly decentralized things. Since these things require diverse computer resources and connect autonomously and situation-specifically to applications, an infrastructure is needed which supports this choreography and which allows to use the computer resources meeting best the requirements of each individual thing. The requirements are specified and further the modules of such an infrastructure are presented. To allow situation-specifically choreography a directory is proposed, which offers thing- and user-specific access to the information needed for the thing to interact. More precisely, the things describe their interfaces and indicate which other things are allowed to access to the respective interfaces.

Since the directory is reachable via a REST API, the things are offered a convenient way to manage their interfaces and the access to them. This way, the participants are independent of the computing resource they are using. Offering gateway services, things without IP-based communication can be included.

With the Identity Provider a single entity is given which issues identity tokens to provide a comfortable authentication at the services, user interfaces and Digital Twins within the network.

Further work will determine the computing resources that the S³I uses. Cloud computing might be convenient for applications where no other computer resource is available that is accessible from all over the respective network. Still, there might be applications that prefer computing resources at the edge level, dealing with privacy issues.

References

1. Aazam, M., Khan, I., Alsaffar, A.A., Huh, E. (eds.): Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved: 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) (2014)
2. Alam, K.M., Saddik, A.E.: C2PS: A Digital Twin Architecture Reference Model for the Cloud-Based Cyber-Physical Systems. *IEEE Access* 5, 2050–2062 (2017)
3. Bierzynski, K., Escobar, A., Eberl, M. (eds.): Cloud, fog and edge: Cooperation for the future? 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC) (2017)
4. Bosch: Bosch IoT Gateway Software: Edge computing for IoT: What is Bosch IoT Gateway Software? (2019), <https://developer.bosch-iot-suite.com/service/gateway-software> (Accessed: 2019-09-10)
5. Boschert, S., Rosen, R.: Digital twin—the simulation aspect. In: *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and their Designers*, pp. 59–74 (2016)
6. Bundesministerium für Wirtschaft und Energie: Plattform Industrie 4.0: Glossar (2019), <https://www.plattform-i40.de/PI40/Navigation/DE/Industrie40/Glossar/glossar.html> (Accessed: 2019-10-31)
7. Eclipse Foundation: Ditto (2019), <https://www.eclipse.org/ditto> (Accessed: 2019-10-31)
8. El-Sayed, H., Sankar, S., Prasad, M., D. Puthal, A. Gupta, M. Mohanty, C. Lin: Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment. *IEEE Access* 6, 1706–1717 (2018)
9. Gulliksson, S., Jones, M., Bradley, J.: OpenID connect Federation 1.0 - draft 04 (2018), <https://openid.net/specs/openid-connect-federation-1.0-ID1.html> (Accessed: 2019-10-31)
10. Kagermann, H., Wahlster, W., Helbig, J.: Umsetzungsempfehlung für das Zukunftsprojekt Industrie 4.0: Abschlussbericht des Arbeitskreises Industrie 4.0 (2013), https://www.bmbf.de/files/Umsetzungsempfehlungen_Industrie4.0.pdf (Accessed: 2019-11-06)
11. Kompetenzzentrum Wald und Holz 4.0: Der Digitale Zwilling in Wald und Holz 4.0: Semantik, Struktur, Perspektiven, Umsetzung: Ein KWH4.0-Standpunkt (2018), https://www.kwh40.de/wp-content/uploads/2019/10/KWH40Standpunkt_Der_Digitale_Zwilling_in_Wald_und_Holz_4.0_Version1.0.pdf (Accessed: 2019-10-27)
12. Reitz, J., Schluse, M., Roßmann, J.: Industry 4.0 beyond the Factory: An Application to Forestry. In: *Tagungsband des 4. Kongresses Montage Handhabung Industrieroboter*, pp. 107–116 (2019)
13. Satyanarayanan, M.: The Emergence of Edge Computing. *Computer* 50(1), 30–39 (2017)
14. Schluse, M., Priggemeyer, M., Atorf, L., Rossmann, J.: Experimentable digital twins—streamlining simulation-based systems engineering for industry 4.0. *IEEE Transactions on industrial informatics* 14(4), 1722–1731 (2018)
15. TongKe, F.: Smart agriculture based on cloud computing and IOT. *Journal of Convergence Information Technology* 8(2) (2013)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.