Object-Oriented Internet Reactive Interoperability

Mariusz Postół[0000−0002−9669−0565]

Institute of Information Technology, Lodz University of Technology, Łódź, Poland
mailto:mariusz.postol@p.lodz.pl

Abstract. Information and Communication Technology has provided society with a vast variety of distributed applications. By design, the deployment of this kind of application has to focus primarily on communication. This article addresses research results on the systematic approach to the design of the meaningful Machine to Machine (M2M) communication targeting distributed mobile applications in the context of new emerging disciplines, i.e. Industry 4.0 and Internet of Things. This paper contributes to the design of a new architecture of mobile IoT solutions designed atop of the M2M communication and composed as multi-vendor cyber-physicals systems. The described reusable library supporting this architecture designed using the reactive interoperability archetype proves that the concept enables a systematic approach to the development and deployment of software applications against mobile IoT solutions based on international standards. Dependency injection and adaptive programming engineering techniques have been engaged to develop a full-featured reference application program and make the proposed solution scalable and robust against deployment environment continuous modifications. The article presents an executive summary of the proof of concept and describes selected conceptual and experimental results achieved as an outcome of the open-source project Object-Oriented Internet targeting multi-vendor plug-and-produce interoperability scenario.

Keywords: Industrial communication · Industry 4.0 · Internet of Things · Machine to Machine Communication · OPC Unified Architecture

1 Introduction

Information and Communication Technology has provided society with a vast variety of new distributed applications. By design, the deployment of this kind of applications has to focus primarily on communication technologies. This article addresses further research on systematic design of Machine to Machine (M2M) communication [9, 11, 19, 21] targeting distributed mobile applications in the context of new emerging disciplines, i.e. Industry 4.0 (I4.0) [20] and Internet of Things (IoT) [3], [22]. New architecture is proposed for IoT solutions designed atop of M2M communication deployed as multi-vendor cyber-physicals systems [20]. The architecture is backed by a proof of concept reference implementation.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

All of the applications designed atop of network communication can be grouped as follows:

- **human-centric** - information origin or ultimate information destination is an operator,
- **machine-centric** - information production, consumption, networking, and processing are achieved entirely without human interaction.

A typical **human-centric** approach is a web-service supporting, for example, a web UI to monitor conditions, and manage millions of devices and their data in a typical cloud-based IoT approach. It is characteristic that, in this case, any uncertainty and necessity to make a decision can be relaxed by human interaction.

Coordination of robots behavior in a work-cell is a **machine-centric** example. In this case, it is essential that any human interaction is impractical or even impossible. This interconnection scenario requires the machine to machine communication (M2M) demanding multi-vendor devices integration. From the M2M communication concept, a broader concept of a smart factory can be derived. In this concept, the mentioned robots are only executive parts of an integrated supervisory control system responsible for macro optimization of any industrial process composed into one whole. This approach is called the fourth industrial revolution and coined as Industry 4.0. It is worth stressing that machines - or more general parts - interconnection is not enough, and additionally, parts interoperability has to be expected for the deployment of this concept.

Examining the information exchange over a network, the first challenge to be faced up is information reusability and security. The second challenge is how to make the mentioned machines interoperable if they are provided by a vast variety of vendors.

M2M communication requires information exchange technology. Unfortunately, information is an abstract knowledge and, hence, cannot be directly processed/transferred using technical means. Fortunately, there is a solution that overcomes this issue, i.e. the information must be represented as bitstreams called data. To make this representation useful in the context of information processing there must be defined semantic and syntax rules called semantic-context. The syntax is used to validate the correctness of the bitstream, and the semantics rules associate meaning to the correct bitstreams.

Going beyond the smart factory realm, a similar concept may be used to make any general-purpose entities interoperable. Finally, we get cyber-physical systems where a variety of entities may be aggregated into distributed information processing solutions. In this case, we are opening the public connectivity domain, which requires a globally scoped infrastructure, i.e. the Internet. Parties interconnected over any network require special precautions that must be taken against malicious users to assure the best possible level of security. It is especially crucial if public resources are used to exchange data. To address this security demand suitable protection methods may be applied to:

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

– **network traffic** - accomplished using intermediary devices to enforce traffic selective availability based on predetermined security rules against unauthorized access,

– **data transfer object (DTO)** - accomplished using cipher algorithms against bitstreams formatted as a message traversing the network and containing process data.

For any generic solution addressing the design of the cyber-physical system, the data holder mobility must be considered as well. Mobile data means that it may come from mobile devices or be generated in unpredictable attachment points. If the data places exposition is arbitrary it means that the data appearance must be recognized and processed as an event. A good example of this scenario is a product (e.g. hygiene goods, cosmetics, drugs, etc) global tracking system - an application domain where the IoT term has been coined [3, 22]. One of the arguments for the IoT is allowing distributed yet interlinked devices, machines, and objects (data holders) to interact with each other without relying on human interaction to set-up and commission the embedded intelligence. In case any kind of mobility has to be considered, the next engineering challenge is dynamic discoverability on the network and the possibility of establishing the semantic and security contexts of the parts composing the IoT application.

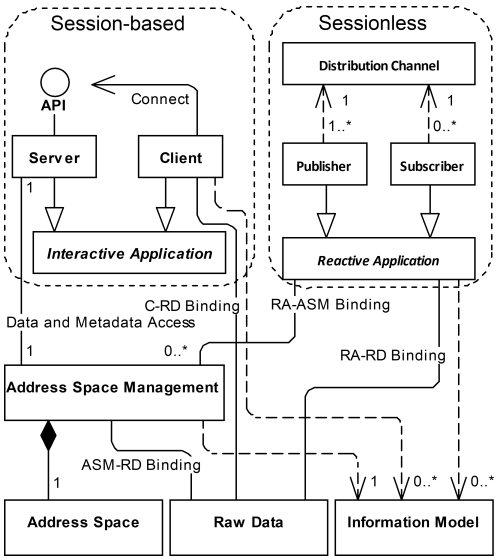

Fig. 1. Interactive/reactive communication

The remainder of this paper is structured as follows. Sect. 2 presents the generic architecture that is to be used as a foundation for further decisions addressing the systematic design of the multi-vendor cyber-physical systems. It focuses on the reusability and security of data processing. In Sect. 3 open and reusable software model is presented. It promotes a reactive interoperability pattern and a generic approach to establishing interoperability-context. A reference implementation of this archetype is described in Sect. 4. The most important findings and future work are summarized in Sect. 5.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

2 Machine to Machine Interoperability

In this section, the most important features of the mobile applications compliant with the more general IoT concept have been abstracted to settle a foundation for a further discussion addressing systematic design, development, and deployment methods and tools. A detailed description of this model is covered by [19]. A starting point for further discussion is a generic architecture of the Machine to Machine (M2M) interoperability presented in Fig. 1. It can be used as a design foundation of the cyber-physical systems. In this approach the following classes have been distinguished:

- *Server* - the front-end component of the server software application,
- *Client* - the front-end component of the client software application,
- *Interactive Application* - a framework supporting the client-server communication pattern,
- *Publisher* - a part derived from *Reactive Application* and implementing *Publisher* role,
- *Subscriber* - a part derived from *Reactive Application* and implementing *Subscriber* role,
- *Reactive Application* - a framework supporting the publisher-subscriber communication pattern,
- *Distribution Channel* - a set of intermediary network nodes interconnected by communication links carrying the published messages to the destination delivery points,
- *Address Space Management* - maintenance services of the *Address Space*,
- *Address Space* - an instance of an address space that can be recognized as a replica of the underling process,
- *Information Model* - a formal model defining the syntax of the exchanged process data,
- *Raw Data* - data origin representing information describing the underlying process state and behavior.

It is essential, that the proposed architecture has no single point where human interaction is possible in compliance with the requirements of the M2M communication concept. In [19] it is recommended how the OPC Unified Architecture international standard may be mapped partially on this architecture to satisfy multi-vendor environment requirements.

M2M communication must assure the reusability od the process data. It could be accomplished as a result of the available communication patterns (Fig. 1):

- **session-oriented** - the data source and ultimate data destinations are tightly coupled by a connection-oriented relationship established between *Server* and *Client* entities,
- **sessionless** - the data source and ultimate data destinations are loosely coupled by *Distribution Channel* carrying the process data over the network.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

The session-oriented client/server archetype is the data exchange scenario that requires establishing, in advance, a session relationship before any process data can be sent over the wire. In this case, the connection-oriented services set up a virtual link making a tight relationship between the communicating parties. The session is responsible for retaining state information about each communicating party for the duration of multiple requests. In this scenario, data sharing is supported because many data destinations may establish an independent session with the *Server* at the same time and, as a result, access the shared data.

The sessionless archetype is a message distribution scenario where senders of messages, called publishers, do not send them directly to specific receivers, called subscribers, but instead, categorize the published messages into topics without knowledge about which subscribers if any, there may be. Similarly, subscribers express interest in one or more topics and only receive messages that are of interest, without knowledge about which publishers, if any, there are. In this scenario, the publishers and subscribers are loosely coupled by *Distribution Channel* filtering, buffering and routing the messages to preselected delivery points. They are decoupled in time, space and synchronization [5].

In the session-oriented communication scenario to establish a new session, the client must send an originating request (*Connect*) over the network to the server exposing relevant data. In some circumstances, it may be difficult or even impossible because of the following (Sect. 1) restrictions related to the Internet-based cyber-physical systems:

- **traffic asymmetry** - intentional limits of the network traffic propagation for the security reasons, for example, enforced by a firewall,
- **mobility** - due to data origin mobility the network node may need to move from one attachment point to another losing its previous endpoint address.

The *Server* - hidden behind the firewall (omitted in Fig. 1 for sake of simplicity) to protect the data origin against malicious users - is an example of asymmetric behavior of the network traffic. It causes that the in front *client* instances cannot efficiently send the originating session request to the server. A similar issue is encountered if the server exposing the process data from the underlying origin is mobile, i.e. the network attachment point could change, and, as a result, the network address is not deterministic. In both scenarios, the sessionless scenario may be employed to address this connectivity limitation issue. It is the main reason for selecting a sessionless pattern to be engaged as a communication foundation to implement M2M interoperability over the Internet.

In [15] the Object-Oriented Internet concept is proposed as a systematic approach to be a foundation of M2M meaningful communication targeting session-oriented archetype. It is derived from well-known Object-Oriented Programming (OOP), which is a paradigm defining objects and their interactions to design computer programs. The goal is to provide a generic solution for publishing and updating information in a context that can be utilized to describe and discover it. Now the possibility to expand this concept addressing the sessionless scenario is researched.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

To make two parties interoperable both must use the same semantic-context to assign the information (meaning) to bitstreams (data) exchanged over the wire (Sect. 1). In other words, there must be a shared understanding of the mutually processed data. In general, the semantic-context may be agreed upon at runtime or design-time, but always in advance. According to the proposed model, this process is supported by the *Address Space*, *Address Space Management*, and *Information Model* concepts, which may be partially implemented [19] in compliance with the OPC UA standard [1, 2, 18].

The runtime approach is straightforward after establishing the session because the session is a tight relationship that can be used to exchange appropriate metadata [15]. In the context of the metadata exposed by the mentioned entities, the client can select what data it is interested in and implement necessary rules to establish the semantic-context using the *Server* services. Because there is no similar tight relationship between the *Subscriber* and *Publisher*, the *Subscriber* must deal with this issue in the reactive rather than proactive way. Lack of standardization in this respect (for example in the OPC UA PubSub [2]) shall be recognized as an interoperability issue that is difficult to overcome in the multi-vendor environment.

Protecting data exchanged over the network requires shared security artifacts that have to change over time to increase the protection strength. It needs a dynamic but stateful relationship between the data source and the ultimate data destination. The session makes the communication parties tightly coupled and therefore may be used as the foundation for establishing also a stateful security-context. In case the sessionless communication pattern is considered the only option is the indirect security-context established using out-of-band communication means. Again, this procedure must be precisely described by an interoperability standard in the multi-vendor environment.

3 Reactive Interoperability Domain Model

As it was pointed out in Sect. 2 the *Subscriber* must deal with data exchange and establishing a semantic-context coupled with security-context in reactive rather than a proactive way. In Fig. 2 a generic domain model of the reactive interoperability archetype is proposed. The *Publisher* and *Subscriber* are derived from a generic *Reactive Application* class, which represents common functionality. They express the publisher and subscriber roles behavior accordingly and fulfill more specific functionality aimed at allowing message centric communication where the primary relationship between process data origin and ultimate data destination is the shared understanding of:

- semantics (meaning) of exchanged process data encoded into and carried by a *Message*,
- the syntax and semantics of *Messages* that include the process data,
- a common *Distribution Channel*.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

In this relationship pattern the *Publisher* is responsible for the encoding of the process data into *Message* entities and for pushing the messages to the *Distribution Channel*. The *Distribution Channel* accomplishes a set of virtual communication routes interconnecting the *Publisher* and *Subscriber* instances. The *Distribution Channel* is a composition of *Intermediary* network nodes interconnected by communication links. The *Intermediary* performs messages multicasting, cloning, filtering, queueing, and forwarding to carry a copy of the *Message* from the *Publisher* to all interested *Subscribers*. This way a virtual path traversed by *Messages* is created. The mentioned functionality is parametrized using *Topic* values and the *Message* entities content attributed by metadata. This asynchronous delivery scenario (colloquially speaking 'fire and forgot') may be recognized as message centric communication.

At preparation time, the subscribers express an interest in one or more topics and, by design, they receive only messages that are of interest, without knowledge about which publishers, if any, there are. As a result, the message must be self-contained and meaningful also outside of the process data origin context. To implement this functionality, the *Subscriber* instances have to parametrize the routing behavior of the *Distribution Channel* after association with *Distribution Channel*. Finally, the messages are pulled and processed by the *Subscriber*. Fulfilling this role the *Subscriber* entities are being reactive. It means that they don't take any initiative or make strategic decisions at runtime. Because in this scenario the *Publisher* uses push operation to forward messages and the *Subscriber* uses pull operation to recover messages, the end-to-end interconnection is limited to one way only. It is worth stressing that two independent communication channels and role coupling are required to obtain bidirectional communication.

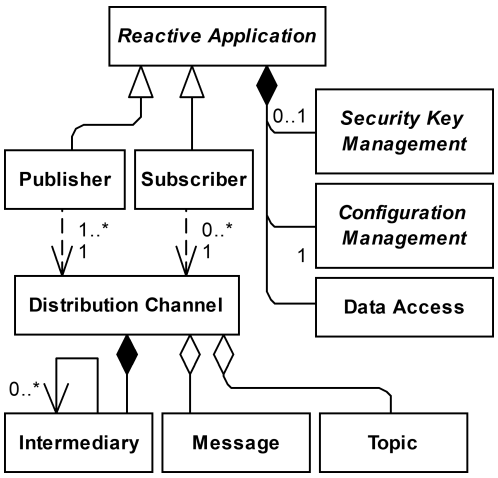

Fig. 2. Reactive interoperability model

The proposed domain model (Fig. 2) allows the implementation of all possible multiplicity relationships between the process data origin and ultimate data destination. For example, *2..* to 0..** relationships are important if data origin redundancy is required. *1..* to 2..** relationships may be considered if the same data should be processed in many locations and/or for a variety of reasons.

In the presented domain model, the *Publisher* and *Subscriber* instances are loosely coupled. It means that their interoperability is not based on a common

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

context established directly between communicating parties similar to a session. Instead, the interoperability semantic-context must be created against the *Topic* values. It makes the *Topic* metadata a vital factor that has to contribute to the behavior of the *Intermediary* nodes and a piece of common knowledge that is required to establish interoperability.

Nevertheless, a prerequisite for establishing interoperability of the communicating parties is an underlying semantic-context that must be created at the preparation step. For the session-based communication pattern, in advance exchange of control messages is applied for this purpose. By design, any bidirectional exchange of messages is unacceptable for the publisher-subscriber communication pattern. A variety of solutions can be applied in this case but all can be derived from one of the following approaches or combination thereof:

- **static** - based on consistent or common configuration files,
- **dynamic** - based on the out-of-band communication between the *Publisher-Subscriber* and *Distribution Channel* instances,
- **remote** - based on the out-of-band communication between an independent configuration server and *Distribution Channel* instance,
- **attributed** - based on metadata (attributes) added to the messages by the forwarding mechanism.

Security between the process data origin and ultimate data destination refers to data protection measures against malicious users. The main tasks of the security mechanism aim at protecting against unauthorized data access, guaranteeing data consistency and non-repudiating while *Message* is transferred by the *Intermediary* nodes. Implementation of any security mechanism requires common knowledge of:

- **cipher algorithms** - mathematical formula necessary to protect the data concerned,
- **keys** - arbitrary streams of bits applied as actual parameters of the cipher algorithms.

If security requires common knowledge it must be deployed based on a mutually shared entity, e.g. session, link, semantic-context, etc. Security integrated with the session is a typical approach for the tightly coupled scenario. Links interconnect adjacent *Intermediary* nodes, but applying link-based security we must deal with a collection of security contexts and hop-by-hop security. This security aggregation approach is recognized as not robust because it is prone to a man-in-the-middle attack. Using semantic-context or establishing an independent context for the security purpose enables the implementation of the end-to-end security between publishers and subscribers that must be a common harmonized activity represented as the *Security Key Management* in Fig. 2. This architecture enables the implementation of the security artifacts management using practically any centralized or distributed technology including but not limited to the blockchain [12].

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

Because the interoperability of the *Reactive Application* is based on the semantic-context shared at the preparation step, the *Configuration Management* services have to be considered as a common activity.

4 Reactive Interoperability Implementation

This section covers an analysis of how to use the proposed domain model (Sect. 3) to make strategic design decisions and distribute functionality to reusable loosely coupled parts using the dependency injection software engineering and adaptive programming using the approach proposed in [16].

The *Reactive Interoperability* concept has been implemented as an open-source library named *SemanticData* in the project Object-Oriented Internet [17] designed to be a foundation for developing application programs that are taking part in the message-centric communication pattern. From the above discussion, we can learn that the main design decisions must concern standardization and flexibility. Standardization needs the selection of an international interoperability specification to make the library ready to be adopted by the multi-vendor environment. Flexibility requires an architecture that promotes the polymorphic independent implementation of essential functions.

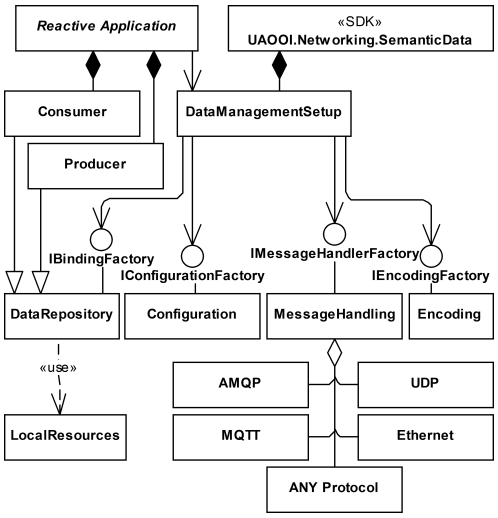

Fig. 3. Reactive interoperability implementation

Piece by piece integration of a cyber-physical system using multi-vendor products requires that M2M communication employs international standards as the interoperability foundation. Following the presented conclusions, OPC Unified Architecture Part 14 PubSub [2] is selected in this respect. By design, this standard should support the required publisher-subscriber communication pattern. Unfortunately, as it is pointed-out in Sect. 3, it covers only partially the requirements of the applications concerned. It must be stressed that by design it provides only an abstract specification. Abstract means that the standard must not limit the implementation strategy. This relationship shall be recognized as the proof of concept to verify that the implementation of the proposed model is feasible to be compliant with the selected standard as envisioned and open to support all functionality required to establish the interoperability context.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

For many parts of the *Reactive Application* domain model (Sect. 3) a polymorphic approach to implementation is required. To promote the polymorphic ready solution the following concepts have been adopted:

- **separation of concerns** - to allow an independent development of the parts [8],
- **dependency injection** - to allow late binding of separately implemented parts [6].

In Fig. 3 the implementation architecture of the *Reactive Application* is proposed. The common functionality has been implemented as the Software Development Kit (SDK) available as the NuGet package [17]. To promote the polymorphic approach, it has the factory class *DataManagementSetup* that is a placeholder to gather all injection points used to compose external parts. To be injected the parts must be compliant with appropriate contracts expressed as the following interfaces representing the following functionality:

- *IBindingFactory* - bidirectional data exchange with the underlying process,
- *IConfigurationDataFactory* - the configuration data access,
- *IMessageHandlerFactory* - pushing the *Message* entities to/pulling from *Distribution Channel* (Fig. 2),
- *IEncodingFactory* - searching a dictionary containing value converters.

It is expected that the functionality implementation represented by these interfaces is provided as external composable parts.

4.1 Process Data Access

The *DataRepository* represents data holding assets in the *Reactive Application* and, following the proposed architecture, the *IBindingFactory* interface is implemented by this external part. It captures functionality responsible for accessing the process data from *LocalResources*. The *LocalResources* represents an external part that has a very broad usage purpose. For example, it may be any kind of the process data source/destination, i.e. *Raw Data* or *Address Space Management* (Fig. 1). By design, the *DataRepository* and associated entities, i.e. *Local Resources*, *Consumer*, *Producer* have been implemented as external parts, and consequently, the application scope may cover practically any concern that can be separated from the core *Reactive Application* implementation.

Depending on the expected network role the library supports the external implementation of:

- *Consumer* - entities processing data from incoming messages,
- *Producer* - entities gathering process data and populating outgoing messages.

The *Consumer* and *Producer* parts are derived from the *DataRepository* (Fig. 3). The *Consumer* uses the *IBindingFactory* to gather the data recovered from the *Message* instances pulled from the *Distribution Channel*. The received data may be processed or driven to any data destination. The *Producer* mirrors the *Consumer* functionality and, after reading data from an associated source, populates the *Message* using the gathered data.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

4.2 Configuration Management

Based on the domain model analysis we can infer that the interoperability of communicating parties requires establishing a semantic-context and security-context between them in advance. The *Topic* based semantic-context is a foundation to ensure a common understanding of the process data encoded into the *Message* instances. On the other hand, the security-context assures mutually shared security artifacts (i.e. cipher algorithms, keys, etc.) used to protect the *Message* as one whole. Both are necessary for handling messages. It is assumed that the security-context is established on top of the semantic-context. In that approach, the data is recognized as a principal of the security measures and the topic is regarded as a name of a collection containing the *Message* instances that are to be protected using the associated security-context independently of the endpoints that produce or consume them.

Therefore, a prerequisite for establishing interoperability of the communicating parties is an underlying semantic-context that must be created at the preparation step. In the proposed implementation, realization of this step is based on the configuration using the implementation of the *IConfigurationDataFactory* interface.

Decoupling of this functionality implementation from the functionality activation using an abstract contract and late binding mechanism allows:

- implementation of practically any configuration management scenario,
- modification of this functionality later after releasing the library or deploying the application program in the production environment.

The preparation phase concerns all the parts composed to make a running instance of the application program using the dependency injection approach to allow separate development and late binding. This approach makes any modification straightforward at any development and maintenance stage. From the end-user point of view, the composition process of the running program using independently developed parts is invisible. Because in any case a single program instance is created in a typical approach, we shall expect a commonly shared configuration mechanism providing mutually exclusive parameters to separately developed parts. This scenario has been addressed by the proposed implementation thanks to making the configuration expandable.

4.3 Distribution Channel Access

Messages preparation and the pull/push operations, which are essential to get access to *Distribution Channel* require an implementation of the interfaces *IEncodingFactory* and *IMessageHandlerFactory*.

The *IEncodingFactory* is used by the SDK to encode and decode the *Message* entities. To make the parties associated with the same *Distribution Channel* interoperable they all must use the same *Message* syntax. Implementation of the *IEncodingFactory* should address one of the options defined in the specification: JSON or binary.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

The *IMessageHandlerFactory* creates object supporting operations: pull incoming messages from and push outgoing messages to adjacent *Intermediary* embedded in an abstract *Distribution Channel* (Fig. 2). To fulfill this task it has to use a concrete protocol stack. The OPC UA PubSub [2] specification lists the following protocol stacks:

- *UDP* - UDP protocol [13] that is used to transport UADP NetworkMessages,
- *Ethernet* - Ethernet-based protocol that is used to transport UADP NetworkMessages,
- *AMQP* - Advanced Message Queuing Protocol (AMQP) [7] that is used to transport JSON and UADP NetworkMessage,
- *MQTT* - Message Queue Telemetry Transport (MQTT) [4] that is used to transport JSON and UADP NetworkMessage.

The library [17] contains a reference implementation of the *IMessageHandlerFactory* for the *UDP* protocol. Because the UDP protocol is used as the *Distribution Channel*, the external filtering of massages is possible only based on the IP address. In this case, the destination port and multicast IP address combination may be recognized as the *Topic* (Sect. 3). Unfortunately, the PubSub specification doesn't provide any mapping outline addressing the question of how to express *Topic* for a particular underlying communication stack. Additionally, it seems difficult or even impossible to create any directory services based on the IP addressing mechanism because it is used for nodes identification and localization on the global network, but not to express data semantics (data meaning). To overcome this limitation it is proposed to use globally unique identifiers of types defined in compliance with OPC UA Information Model [10, 14] as the *Topic* entities to establish semantic-context and security- context of the reactive assets interoperability. In this approach, the filtering and multiplexing functionality must be embedded locally in the implementation of the UDP communication stack.

It is worth stressing that the proposed separation of concerns and dependency injection approach make the architecture ready to utilize *ANY Protocol* that supports transparent data transfer over the wire (Fig. 3).

5 Conclusions

Based on the architecture proposed in [19] and abstracted in Sect. 2 the sessionless and session-oriented communication patterns are examined against the IoT requirements reviewed in Sect. 1. The discussion concludes that the connectionless pattern better suites issues related to the assets mobility and traffic asymmetry that is characteristic for the application domains concerned. Additionally, to promote interoperability and address the demands of the M2M communication in the context of a multi-vendor environment the implementation of products derived from the proposals must be compliant with the selected international standard. The mapping of this architecture and OPC UA is discussed in [19].

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

In Sect. 3 a generic domain model of the reactive interoperability archetype is introduced. It has been designed based on the above-mentioned findings. The main goal is to provide a foundation for the future development of standards, best practice rules and supporting reusable frameworks. It was pointed out that improvements of the existing interoperability standards addressing the reactive interoperability based on the publisher-subscriber archetype, e.g. OPC UA PubSub [2], AMQP [7] and MQTT [4] shall be scoped on establishing the semantic and security contexts. Considering best practice rules it is worth stressing that the process of establishing an interoperability context is based on the firm but abstract rules. However, the concrete implementation must be flexible enough to deal with a variety of polymorphic algorithms. Addressing the development of a reusable framework needs the proposed model to be backed by proof of concept, i.e. a reference implementation described in Sect. 4.

The main aim of Sect. 4 is to present the experience gained during the implementation of the reactive interoperability concept outlined in Sect. 3. This concept was implemented consistently with the Object-Oriented Internet paradigm[1] [17] worked out in an open-source project. The domain model proposed in Sect. 3 is used to make significant implementation decisions. The description of a reference application program implementation proves that it is possible to design universal architecture targeting reactive interoperability as a consistent part of the Object-Oriented Internet concept compliant with the OPC UA PubSub [2] international standard. According to the presented implementation and evaluation, using the dependency injection and late binding, the application program can be seamlessly adapted to the production environment and scales well.

The results presented in the article prove that the composite nature of the disciplines concerned can be relaxed by the composite nature of the running programs and postponing parts binding up to the system deployment stage. It also improves flexibility and adaptability of the existing solutions against any modification of the production environment including but not limited to the selected interoperability standard change.

Future work is focused on Machine to Sensors connectivity based on Process-Observer concept introduced in [15]. By design, it allows access to plant floor devices using a variety of Fieldbus industrial network protocols. The main goal of this project is to prove that, based on the presented results, the fetching data from Process-Observer will make the factoring of the structural data as the composition of simple data possible.

References

1. Opc unified architecture specification part 3 – address space model (2017), https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-3-address-space-model/
2. Opc unified architecture specification part 14 - pubsub (2018), https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-14-pubsub/

[1] https://github.com/mpostol/OPC-UA-OOI

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31

3. Ashton, K.: That 'internet of things' thing. RFID JOURNAL **2009**, 1–1 (Jun 22, 2009), https://www.rfidjournal.com/articles/pdf?4986
4. Cohn, R.J., Coppen, R.J.: Mqtt version 3.1.1 plus errata 01 (10 December 2015), http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
5. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Computing Surveys **35**(2), 114–131 (2003). https://doi.org/10.1145/857076.857078
6. Fowler, M.: Inversion of control containers and the dependency injection pattern (23 January 2004), https://www.martinfowler.com/articles/injection.html
7. Jeyaraman, R., Telfer, A.: Oasis advanced message queuing protocol (amqp) version 1.0 (29 October 2012), http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html
8. Kulkarni, V., Reddy, S.: Separation of concerns in model-driven development. IEEE Software **20**(5), 64–69 (2003). https://doi.org/10.1109/MS.2003.1231154
9. Lawton, G.: Machine-to-machine technology gears up for growth. Computer **37**(9), 12–15 (2004). https://doi.org/10.1109/MC.2004.137
10. Mahnke, W., Leitner, S.H., Damm, M.: OPC Unified Architecture. Springer, 1 edn. (May 2009)
11. Meng, Z., Wu, Z., Muvianto, C., Gray, J.: A data-oriented m2m messaging mechanism for industrial iot applications. IEEE Internet of Things Journal **4**(1), 236–246 (2017). https://doi.org/10.1109/JIOT.2016.2646375
12. Novo, O.: Blockchain meets iot: An architecture for scalable access management in iot. IEEE Internet of Things Journal **5**(2), 1184–1195 (Apr 2018). https://doi.org/10.1109/JIOT.2018.2812239
13. Postel, J.: User datagram protocol (28 August 1980), http://www.ietf.org/rfc/rfc768.txt
14. Postol, M.: OPC From Data Access to Unified Architecture, sec. Information model, pp. 111–130. VDE VERLAG GMBH, 4th revised edition edn. (2010)
15. Postol, M.: Object oriented internet. In: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS). pp. 1069–1080 (2015). https://doi.org/10.15439/2015F160
16. Postol, M.: Csharp in practice adaptive programming. Tech. rep., Lodz University of Technology (February 26, 2019). https://doi.org/10.5281/zenodo.2578244
17. Postol, M.: Object-oriented internet. Tech. Rep. 5.1.0 (2019). https://doi.org/10.5281/zenodo.3345043
18. Postół, M.: Opc ua information model deployment. Tech. rep., CAS (April 18, 2016). https://doi.org/10.5281/zenodo.2586616
19. Postół, M.: Computer Game Innovations 2018, chap. Machine to Machine Semantic-Data Based Communication: Comprehensive Survey, pp. 83–101. Lodz University of Technology Press, Łódź Poland (2018), https://www.researchgate.net/publication/335524620_Computer_Game_Innovations_2018
20. Schlick, J.: Cyber-physical systems in factory automation-towards the 4th industrial revolution. 9th IEEE International Workshop on Factory Communication Systems (WFCS) (2012)
21. Verma, P.K., Verma, R., Prakash, A., Agrawal, A., Naik, K., Tripathi, R., Alsabaan, M., Khalifa, T., Abdelkader, T., Abogharaf, A.: Machine-to-machine (m2m) communications: A survey. Journal of Network and Computer Applications **66**, 83–105 (2016). https://doi.org/10.1016/j.jnca.2016.02.016
22. Xu, L.D., He, W., Li, S.: Internet of things in industries: A survey. IEEE Transactions on Industrial Informatics **10**(4), 2233–2243 (2014). https://doi.org/10.1109/TII.2014.2300753

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_31