Enabling Smart Manufacturing by Empowering Data Integration with Industrial IoT Support

Enabling Smart Manufacturing by Empowering Data Integration with Industrial IoT Support

Filippo Bosi[2], Antonio Corradi[1], Giuseppe Di Modica[1], Luca Foschini[1], Rebecca Montanari[1], Lorenzo Patera[1], Michele Solimando[1]

[1]Department of Computer Science and Engineering, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
{antonio.corradi, giuseppe.dimodica, luca.foschini, rebecca.montanari, lorenzo.patera, michele.solimando2}@unibo.it

[2]Imola Informatica SpA, Via Selice 66/A, 40026 Imola (BO), Italy
fbosi@imolainformatica.it

***Abstract*—Industry 4.0 (I4.0) denotes the start of a new era where another industrial revolution is urged to explode. Simply put, I4.0 aims at enhancing the competitiveness in the manufacturing industry, while at the same time improving the safety and security of work centers. The achievement of that goal is bound to the adoption of enabling information technologies (IoT, Cloud, Big Data) to build a unified and integrated view of the information generated at all stages of the industrial processes, both operational and business oriented ones. This paper focuses on the need for manufacturing companies of tools that concretely implement this data integration as a requirement to enforce the improvements fostered and needed by the I4.0 revolution. This paper follows a former work that proposed the design and implementation of a platform realizing the integration of the operational and the business views at the data level. We enhanced the existing platform features by introducing new component that implements the interface with IoT devices via well-known communication protocols, namely MQTT and AMQP. Experiments run to test the newly introduced feature show the viability of the proposed approach with respect to the time constraints that are typical of a manufacturing production environment.**

***Index Terms*—Industry 4.0; Smart Manufacturing; RAMI 4.0; SCADA; OPC-UA; Cyber-Physical Systems; Cloud-enabled Data Collection; MQTT; AMQP**

I. Introduction

Industry 4.0 (I4.0) revolution is bound to yield a remarkable change to the way industry approaches the manufacturing area. Mature Information Technologies (IT) such as IoT, Cloud Computing and Big Data can sustain the concept of "industry digitization". This pervasive transformation process will force companies to develop a unified data infrastructure where information generated at operation level (i.e., produced by machines, production chains, etc.) integrate with both end-user data and business data. This integration will require to make the great amount of operation level data available at the enterprise level. In the light of that, companies will have better insights on the live production performances and will be able to undertake both short-term and long-term business actions. Among the benefits of interoperation between the operational technology (OT) layer and IT layer, one can certainly list better machine utilization, better performing production lines, enforcement of predictive machine maintenance, improved security for people working alongside the machines, opportunity to run new business models, and many more.

As I4.0 aims to connect all parties involved in manufacturing processes, a strong need for reaching a broad consensus over the semantics of the prospected context has arisen. Many standardizing initiatives have been promoted in several countries that all share the common goal of laying down reference models and technical specifications to drive the aforementioned revolution [1]–[6]. Among those initiatives, we focus on putting great emphasis on the OT/IT convergence. Along that direction, the Reference Architecture Model for Industrie 4.0 (RAMI 4.0) [1] is a European initiative that defines a service-oriented architecture, leveraging different technological solutions to address I4.0 issues. As for low-level data acquisition and communication RAMI 4.0 embraces, among others, the well known Supervisory Control and Data Acquisition (SCADA) [3], a communication protocols family for legacy devices, and the Open Platform Communications - Unified Architecture (OPC-UA) [7], and industrial M2M communication protocol designed with interoperability in mind.

Though many industrial reference models are covered by the big umbrella of RAMI, full data integration and OT/IT interoperability is barely addressed. Guided by the need to fill this gap, in a former paper we proposed the definition of a cloud-enabled architecture. Our architecture supports data management between the OT and IT levels in an interoperable manner and to monitor legacy production machines during their

operations inside customers plants. This paper enhances that paper's achievements. First, it provides the platform architecture with a novel component that implements the support for IoT communication protocols MQTT and AMQP; second, it implements a software prototype by leveraging well known open source tools; third, it provides the needed support to meet mission critical requirements typical of cyber-physical working environments; fourth, it presents experimental results that confirm the viability of the proposed solution.

The rest of the paper is structured as follows. Section II glances on the I4.0 standardization initiatives that inspired this work. In Section III the platform's design and implementation principles are recalled, while in Section IV the result of the tests are presented and commented along. Related work is discussed in Section V. Finally, Section VI concludes the work.

II. Background

In the last few years I4.0 has attracted many private and public funds. The manufacturing sector is strategic for new generation industries in which customers can buy customized products, while optimized data and control flows permit to efficiently drive the plant. Advanced control flows are fed by multiple IoT sensors that collect information from the plant and forward them to entities that analyze and recommend the necessary control actions. The amount of data that in an industry is daily gathered and analyzed is huge. Every sensor can sample and transmit data at a very strict time interval and with different granularity. Usually, data communication is not one-to-one, since multiple actors can be interested in the same data flow. Moreover, protocols used by Industrial IoT (IIoT) sensors are diverse and very heterogeneous, thus normalization actions over data flows are often necessary.

In this section, we provide a view of the main existent platforms and standards concerning I4.0. Without any pretense of being exhaustive, the goal of the section is to facilitate the identification of the critical aspects and stress the differences with respect to the proposed solution (see Section III). We present three emerging standards. The first one is the Reference Architectural Model Industrie 4.0 (RAMI 4.0) [1], a three-dimensional spaced model defining how to approach I4.0 issues in a structured manner. Thus, we explore Supervisory Control and Data Acquisition (SCADA) [3], a technology widely is being used in industrial and in manufacturing plants from over 30 years. Finally, we investigate the Open Platform Communications (OPC) protocols [8], which focuses on secure and real-time machine-to-machine (M2M) interoperable communications.

A. Reference Architecture Model Industrie 4.0

In Figure 1, RAMI 4.0 is depicted. The model ensures that all the entities involved in the platforms

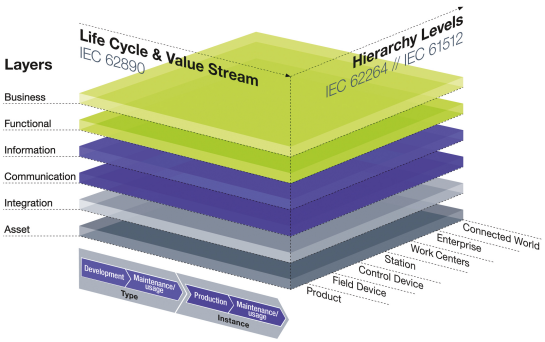

Fig. 1. Reference Architectural Model Industrie 4.0

can communicate in a uniform and standardized within a service-oriented architecture. It makes wide use of the *divide and conquer* principle, splitting the complexity in several packages, including data privacy and IT security. The model develops in three distinct yet complimentary dimensions:

- **Hierarchy Levels** (IEC62264/IEC61512). This axis models the environment surrounding the industry. It spans from the product to the perspective of a connected world, opening the system to other external enterprises, devices and smart things in general. It is compliant with the 62264 and 61512 IEC standards. The former is an enterprise standard for system integration having its roots in the ANSI/ISA-95 [9] international standard: it helps to define boundaries between the enterprise systems and the control systems. The latter defines reference models for the batch control (as it is used in the process industries) and the terminology explaining the relationships between these models and the terms.
- **Life Cycle & Value Stream** (IEC62890). This axis identifies two main phases. The first one (*Type*) defines the entry of design, development and test orders, carried out up to the first sample and the production of the following prototypes. At this stage, therefore, the type of product, machine, etc. is represented. Only at the end of all tests and the corresponding validations, the type is certified and released for series production. The second phase (*Instance*) identifies the products manufactured according to the general type described in the previous phase. Each product represents an instance of a specific type and has a unique serial number. Then the requests are transferred and delivered to customers. On the customer side, the products are initially just types. They become instances when they are installed in a specific aggregate system.
- **Layers**. This axis models the classical partition of the Cyber-Physical Systems (CPS). At the bottom the physical things in the real world are

represented (sensors, actuators, etc.). The physical object interfaces directly with its digital representation (*Integration layer*). The digital representation is then shared with the surrounding entities through the *Communication layer*. From the data it is now important to extract information, dependently to the nature of the asset. Then the systems can be integrated in a unique *Functional layer*, providing input for the top *Business layer* which enables the development of new business processes and better organization of the plants.

B. Supervisory Control and Data Acquisition

SCADA [3] systems are centralized entities devoted to control sensors, actuators and assets, and to trigger corrective actions over them. SCADA systems are often equipped with advanced Graphical User Interface (GUI) that helps operators monitor the plant. The control actions over the plant can either be done by the system or manually driven by an operator. In SCADA systems the controlled plant can be highly sparse in space and the assets need to guarantee a connection with the SCADA servers in order to be monitored and controlled. The assets are often called *field devices*, since they act directly on the plant with operations such opening and closing valves and breakers. The manners in which SCADA application interact and control the assets are several. Usually, every SCADA-compliant component has to expose an accessible protocol, such as MODBUS [10] or a carrier such as the MQTT [11] or the AMQP [12] protocols, through which the asset state can be monitored. Moreover, SCADA usually hides the underlying protocols complexity, giving the illusion a unique interface of interactions.

C. Open Platform Communications

OPC operates in the context of devices intercommunication in a client-server manner. Its main purpose is a unique and standardized way to easily and securely exchange data between different industrial platforms from multiple vendors. The OPC products are more than 35.000 and the specifications help to overcome interoperability issues and to eliminate the need of post-production standardization efforts [13]. Also, there are several OPC specifications. The Classic one [8] derives from the Microsoft Distributed Component Object Model (COM/DCOM) [14]. The whole set of protocols is broken down into three main categories, according to the type of data that can be accessed: Data Access (DA), Alarms and Events (AE) and Historical Data Access (HDA).

With the advent of the Service Oriented Applications (SOA) approach [15], in 2008 OPC specifications too evolve into a more complex and powerful architecture: the OPC Unified Architecture (OPC-UA) [7]. Similarly to RAMI 4.0, OPC-UA adopts a multi-layered approach that targets the I4.0 emerging problems. The architecture is functionally equivalent to the Classic OPC specification and is also platform independent, secure and extensible.

III. Platform design and implementation insights

In this section we illustrate the design and the implementation details of the proposed layered platform. We designed the platform architecture to overcome some industrial integration issues, but also to be scalable and to correctly address the different stakeholders' necessities. Basically, the architecture's purpose is to serve the need of extracting real-time data from heterogeneous industrial machines and to present stakeholders (plant operators, IT expert, machine vendors) with personalized subsets and aggregated information views from the plant. The platform stems from tight collaborations with important manufacturing industries that contributed to identify the company's operational requirements, safety and security issues. Figure 2 shows an overview of the entire architecture. The architecture components are deployed in different locations. Each component has its own constraints in terms of communication delay, data granularity and security. As mentioned before, the architecture follows the RAMI 4.0 layers specification. Further, it takes into consideration the OPC-UA specifications in that it copes with legacy or new protocols at the OT layer. With regards to the data extraction and transformation approach, the reader will find some similarities between our proposal and the one discussed in [16]. Starting bottom up, the architecture consists of 5 layers: Machine Layer, OT, Mirroring Layer, IT and Cloud. In the following, we glance on each component details and on the open source tools that were used at implementation time. In the following figures, each tool's color matches that of the architecture's component it implements (see Figure 2).

A. Machine Layer

The machine layer is characterized by cyber-physical systems (CPS) that command machine actuators and control the correct behavior of the systems. Those components can be very complex and must meet safety and security regulations, as they can injure workers if compromised. Furthermore, the CPS act in real time on actuators, striving to meet the maximum request-response delay constraints which are in lower order of magnitude than to those of the IT systems. In order to interact with the CPS, many machines expose interfaces such as MODBUS and PROFIBUS [17] from the SCADA family, OPC-compliant devices, MQTT or AMQP IIoT middleware platforms. Being compatibility with legacy systems one of the I4.0 priorities (not all companies can sustain the IT turn

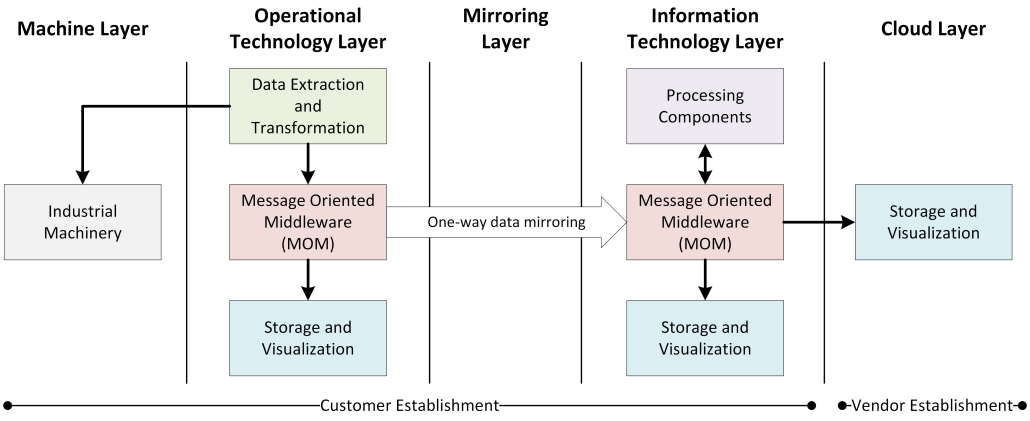

Fig. 2. Overall architecture schema

over by periodically getting rid of legacy machines), integration platforms must deal with existent machinery and be prepared to integrate future protocols. Finally, machine communication protocols are very low-level, thus the system will deal with the conversion from the binary raw data produced by machines to a high-level usable representation.

B. Operational Technology Layer

This layer represents the first and unique data entry point of in our architecture. It lays on top of the Machine Layer and interfaces directly with it. It includes several components, and is entrusted with the extraction and normalization of data as well as with their middle-term storage. Moreover, since it has direct access to the machine controllers, and many CPS protocols do not have advanced access control policies, it must be placed in a network domain having the same high security level as the machines. Figure 3

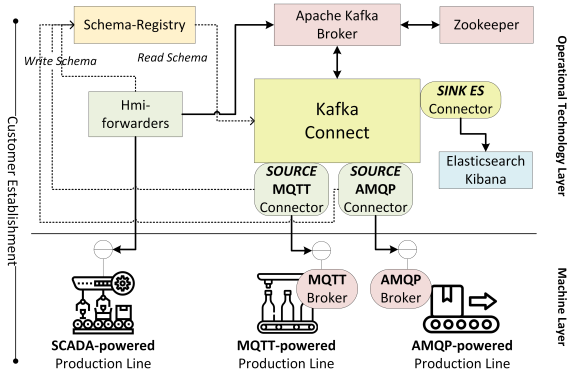

Fig. 3. Operational Technology Layer schema

shows a more detailed schematic of this layer. In the following subsections, we deeply explore main OT-layer components and their main roles.

1) Hmi-forwarders: Human Machine Interfaces (Hmi) forwarders are committed to extract data from the SCADA-powered machinery, and to normalize and forward them to the Kafka Broker. We split the component's responsibilities into three main roles: extracting data , casting the type to the corresponding application value (int, float, etc.), converting, serializing and sending to the broker. The first part is strictly SCADA-protocol dependent and can be performed in push mode in advanced systems or via polling in legacy machines. Often, data taken from the machine registers are expressed in non-standard units (e.g. square instead of meters) and must be converted in order to be uniformly interpreted by other system parts. The serialization is crucial when dealing with structured data, as it permits to compress the messages and to give them a fixed structure. In our system this is achieved with a schema-based approach. The Hmi-forwarder takes care of storing a schema of the data in the Schema-Registry [18] component. The Schema Registry is a server component developed by Confluent Inc. that provides RESTful interfaces for storing and accessing AVRO [19] schemas in a reliable manner. The Hmi-forwarder then serializes the data according to the stored schema and sends a message to the broker containing both the data and the id of their schema, which is required for deserialization purpose. Each Hmi-forwarder publishes messages in just one topic of the message broker. We stress that Hmi-forwarders can be even more than one per machine, depending on the machine's registers composition and on the logical splitting of the data adopted by the message broker.

2) Kafka: Core component of the layer is the Message Oriented Middleware, which is implemented via the Apache Kafka tool [20]. It allows to decouple message senders (producers) from messages receivers (subscribers). Main abstraction of Kafka is the *topic* concept. A topic is a feed name to which messages are stored and published. It is composed of several *partitions* that are immutable and ordered set of messages. Each partition can be replicated in several Kafka

Broker instances in cluster mode, so that an occasional broker failure can easily be recovered without message loss. An instance of Zookeeper [21], a hierarchical key-value storage platform, serves the Kafka need to store in a reliable manner reading indices and topic names.

3) Kafka Connect and connectors: Kafka messages can be either consumed or produced directly via the Kafka Connect [22] tool. The latter abstracts even more from the basic pub/sub mechanism, by offering the concept of messages stream and an extensible plugin system for interconnecting the broker with external data sources. Plugins can either be Source (input) connector type or Sink (output) connector type. Each connector instance is composed of a set of tasks that actually copy the information between the platforms. In order to reach the desired level of parallelism, there can be multiple tasks per each connector. In our architecture, we have used Source plugins to interface with AMQP and MQTT, and Sink plugin to export data to the Elasticsearch database. The former allowed us to import data directly from machines powered with the two message brokers without further customization. Connectors can be also configured to operate simple conversions when they transfer data, in order to grant the same capabilities of the custom Hmi-forwarders.

4) Elasticsearch and Kibana: The Elasticsearch and Kibana tools are part of Elasticsearch, Logstash, and Kibana (ELK) stack, which also includes a log management tool. Elasticsearch [23] is a distributed database optimized for real-time queries. It is based on Apache Lucene and can store a wide variety of data, from geospatial information to plain application's structured or unstructured values. Kibana [24] is a front-end application that provides data visualization and search capabilities. It allows the user to create custom real-time views of the data inside Elasticsearch, which are then rendered by means of charts, pie and tables from which industrial experts can monitor the business navigation. We want to stress that those components are examples of potential data consumers of the platform. We believe that both a storage and a real-time view layer in the OT layer of the system are mandatory. They provide machine's operators with information about machine processes in a real-time manner, so that are able to promptly react in case of need.

C. Mirroring Layer

The Mirroring Layer is responsible for interconnecting the OT and the IT layer. It is a crucial component, since it has functional, security and separation responsibilities. The Mirroring layer software subscribes to Kafka topics of the OT layer and produces messages to Kafka topics in the IT layer. The component can be customized to copy only a subset of the topics or even a subset of the messages flow inside a topic. In fact, some topics may contain data representing secret recipes that should be kept protected also within the company (e.g. the sales office). We opted for Apache MirrorMaker (a built-in Kafka component) to serve this purpose. We stress that the two Kafka Brokers are not configured in cluster mode, as otherwise the isolation between the two layers would be violated and all data would flow from OT and IT and vice-versa. MirrorMaker keeps in synch also the schemas by way of a dedicated topic named *_schemas* of which the OT schema-registry is responsible for posting updates.

D. Information Technology Layer

The IT layer accounts for the data analysis needs of the factory. It provides for a security domain where aggregate, pre-process and analyze data. The rationale behind that is to also have a portion of the system (OT) in which all the data remain certified and do not undergo any processing, so that in case of failure of the upper layers the system can still be efficiently restored. With respect to OT layer, the constraints of IT layer are more relaxed, since a potential attack cannot directly harm either workers or manufacturing machines. This layer replicates almost every component of the OT layer (except the *Hmi-forwarders*), and contains processing elements. In Figure 4 a schematic view of

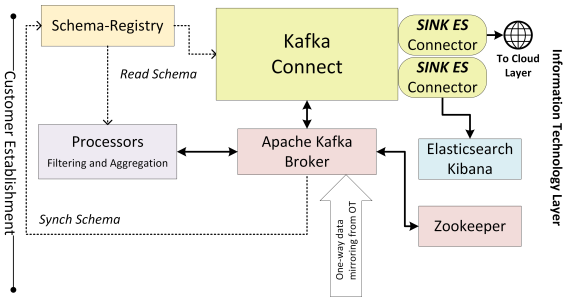

Fig. 4. Information Technology Layer schema

the IT layers components and its interconnections is displayed. We implemented processors using the *Kafka Process* APIs, that allow to define custom functions to be executed on ongoing messages flow. There are also a storage and view layer for the employees that do not directly act on machines, which provides high-level information of the plants.

The components discussed so far are all installed in the Customer establishment. Since third party industries may also be interested in production data, we devised a standard interconnection mechanism. Again, via the Kafka Connect plugins it is possible to export data to a number of external platforms. Topics can be selected and pre-processed, so that the Customer will be able to show a customized view to a third party. To make a concrete example, we figured out

that the vendor of a machine used in the Customer establishment is potentially interested in taking a look at the machine data. Those, in fact, might reveal the health status and the real time performance of the machine. We employ an additional Elasticsearch Connector to export machines data into the so-called Cloud Layer, which in the example is under the control of the machines vendor.

E. Cloud Layer

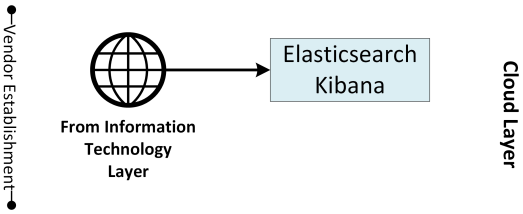

Fig. 5. Cloud Layer schema

The Cloud Layer collects data coming from customers' establishments. This is where advanced data analytics are elaborated. Here, we can find ETL tools and advanced cross-customer analytics tools that help vendors improve the quality of their machines, but also to identify issues in running machinery. We choose again Elasticsearch and Kibana for their simplicity of deployment and testing, but we stress that any storage and analytical tool can be attached by deploying its connector in the underlying layers. In Figure 5 a schematic of this layer is depicted.

IV. Test and Results

In this section, we summarize the results of some integration tests run on the platform. The aim of these tests is to verify the correct behavior of the system interconnected with production lines equipped with MQTT and AMQP. We evaluated two main aspects: stressing the system and assess its scalability.

Focusing on the software prototype, we implemented the architecture described in the Section III. We implemented every architecture component as a Docker container to gain the benefits of lightweight virtualization (ease of deployment and software maintenance). Furthermore, we chose Kuberbetes [25] for orchestrating the containers, and Rancher [26] to manage the Kubernetes clusters.

The testbed consists of three VMs, each equipped with Ubuntu 18.04.01 LTS operating system, 6GB RAM, 100GB hard drive, 6 logical cores and a fiber connection up to 1GBps. We deployed the OT layer components and the MirrorMaker on one VM, while the second VM hosted the IT layer components. Finally, we put the Cloud layer components on the third VM. Though tests were conducted on a small testbed, the platform is cluster-ready and easy to scale by just adding more computational power. We emulated machinery, and in general all physical assets, via software. As for the machine registers, we configured them to exchanged messages of a size ranging from 2.5 KB (simple registers) up to 3.5 KB for complex information aggregating data from multiple registers. The objective of the test is to stress the system capability to cope with intense message workload when the MQTT and the AMQP brokers are respectively employed. In each experiment, which lasts about 16 minutes, we switch on the machines and the registries starts sending messages. Each machine is capable of sending the brokers 24 messages every 30 seconds. For both experiments, the number of machines is constantly incremented, at regular time intervals, from 4 up to 30. We put the focus on two metrics: the number of messages successfully delivered to the view layer and the average end-to-end message delay.

In the depicted scenario, we tested the MQTT performance by deploying an open-source MQTT implementation called Mosquitto [27]. We configured the simulated assets to publish their data on Mosquitto topics, exactly as happens with machines that natively uses MQTT implementations for their internal communications.

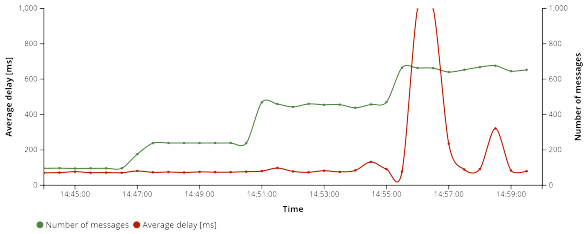

Fig. 6. Delay test using Mosquitto Connector

The red line of Figure 6 shows the trend of the delay measured when the messages are taken from the Kafka broker. The green line represents the number of messages that are correctly dispatched by the MQTT broker. The test proved that the system has an acceptable delay (average value less than 100ms), even when the message traffic increases. However, the reader may notice that at time 14:55, when 30 machines are operating and a message production rate of 720 per 30 seconds is observed at the input of the MQTT broker (1440 messages/minute), the rate of correctly delivered messages decrease and the message delay becomes unacceptable, with a peak of over 1 second. At this stage, with the current deployment, the MQTT broker queue reaches a memory saturation, while no particular misbehaviour of the Kafka brokers was observed.

Similarly to what was done with MQTT, we tested the platform with the RabbitMQ broker [28], the most widely deployed open-source AMQP implementation.

We carried out the experiment keeping the same loads used for MQTT. As showed in Figure 7, the AMQP

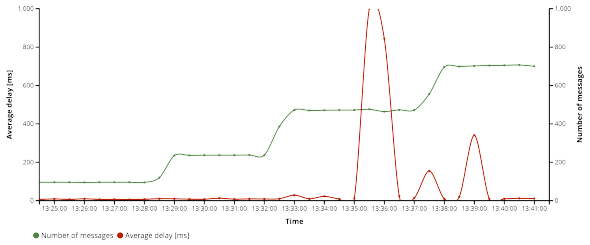

Fig. 7. Delay test using RabbitMQ Connector

protocol outperforms the MQTT's. AMQP shows an average delay of only 14 ms, due to its optimized usage of memory resources. Even though at time 13:35 a peak in message average delay is observed, AMQP manage to deliver the 100% of messages. The peak in the delay is due to the background activity of memory cleaning tasks that are periodically run by the broker.

At the end of the test, we can conclude that when a low-medium message load is injected in the system, the performance in terms of message delivered and message delay are compatible with those required in mission-critical contexts. Unfortunately, the performance degradation observed at a given point in time in both AMQP and MQTT experiments are due to structural limits of the software employed in the tests. A better configuration of AMQP and MQTT parameters and the employment of more powerful VM would have avoided the misbehaviour.

V. Related work

As anticipated, one of the main issues in industrial environments is the integration between different vendor-specific protocols and the overlying SCADA applications. The literature is full of proposals that try to implement RAMI 4.0 standard via OPC integration or via the usage of Message Oriented Middleware [29] to decouple senders and receivers of data. In this section we briefly discuss literature contributions coping with both legacy systems and IoT protocols respectively.

[30] proposes an Open Middleware for industry based on a MQTT broker to gather data from the machines. The proposed solution is not disruptive and permits to fast develop industrial applications. The work stresses the importance of having a dedicated middleware in IoT applications, and stresses that MQTT-based middleware is becoming the most preferred protocol for M2M and IoT scenarios. The Constrained Application Protocol (CoAP) [31] is not sufficient for new generation appliances, that need to rely on pub/sub communication patterns and on affordable transport protocol capable of re-transmitting data in case of network failure.

[16] stresses the need of integration between Operational Technologies (OT) and Information Technologies (IT), and proposes an interoperability layer between them. The layer remaps data into a Common Information Model based on the ISA95 [32] industrial standard for legacy systems. The model is composed of three major parts. The *Raw Data Importer* extracts data from machine registers via low-level specific protocols (such as MODBUS). After the extraction, data in JSON notation are passed on to the *Mapper* component. The Mapper uses the ISA95 as a reference model for describing assets and enriches the data with semantic tags. Enriched data flows from the Mapper to the *Information Provider* that is an OPC-UA server integrated with ISA95 information model. Finally, data are exposed in a standard way and can be accessed via any OPC-UA client.

[33] proposes a platform for collecting and processing of real-time factory shop floor streams of data. The authors adopts a classical IoT approach providing a hub and a gateway to connect devices. The required message exchange is done within state of the art technologies and protocols, e.g., MQTT protocol and REST-based interface. A prototype of the described approach was implemented, deployed and tested in an industrial-based scenario.

The works presented above stress the necessity of having software solutions capable of a) providing a standard way of accessing normalized data from the IT layer and b) being equipped with many specific tools capable of gathering data from OT layer. Furthermore, the industrial revolution also advocates smart legacy machinery integration via message oriented middleware and dealing with the issues affecting the many-to-many communication protocols. However, none of the discussed works devises a layered solution capable of splitting the complexity of the industrial environments from the very bottom machine layer to the upper business layers, addressing the needs of customization and protocol flexibility. Our work follows the RAMI 4.0 guidelines with respect to near real-time monitoring requirements and tackles the industrial division between OT and IT. The latter is not only applied to low-level protocol interactions, but also to conceive a data visibility approach that meets the stakeholders' different needs.

VI. Conclusion

The Industry 4.0 revolution is expected to bring new business opportunities to manufacturing companies, provided that the latter are keen to enact a fast digitization of their assets supported by enabling technologies like IoT and Cloud, to name a few. Though some standardization initiatives have led the way on how such a transition ought to occur, there is only a few proposals of concrete systems and tools that can get

the industry ready for the digitization leap. The data integration platform discussed in this paper enables the industry to communicate with the IoT world, and proves to be capable of meeting the real-time requirements that are typical of manufacturing work spaces. Future work will explore non technical issues of OT/IT integration, with focus on data security in a context of multiple stakeholders.

REFERENCES

[1] "Alignment report for reference architectural model for industrie 4.0/ intelligent manufacturing system architecture," last visited in Jan. 2020. [Online]. Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/hm-2018-manufacturing.html

[2] "Unified architecture part 1 overview and concepts," last visited in Jan. 2020. [Online]. Available: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts/

[3] K. Stouffer and J. Falco, "Guide to supervisory control and data acquisition (scada) and industrial control systems security," 2006.

[4] "Industrial internet reference architecture," last visited in Jul. 2019. [Online]. Available: https://www.iiconsortium.org/IIRA.htm

[5] Y. Lu, K. Morris, and S. Frechette, *Current Standards Landscape for Smart Manufacturing Systems*, 2016, vol. 8107, no. April. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8107.pdf

[6] Q. Li, H. Jiang, Q. Tang, Y. Chen, J. Li, and J. Zhou, "Smart manufacturing standardization: Reference model and standards framework," in *On the Move to Meaningful Internet Systems: OTM 2016 Workshops*, I. Ciuciu, C. Debruyne, H. Panetto, G. Weichhart, P. Bollen, A. Fensel, and M.-E. Vidal, Eds. Cham: Springer International Publishing, 2017, pp. 16–25.

[7] "Unified architecture," last visited in Jan. 2020. [Online]. Available: https://opcfoundation.org/developer-tools/specifications-unified-architecture

[8] "Classic," last visited in Jan. 2020. [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-classic/

[9] A. ISA, "Isa-95.00. 03-2005 enterprise control system integration part 3: Activity models of manufacturing operations management, isa-the instrumentation," *System, and Automation Society*, 2005.

[10] "Modbus 101 - introduction to modbus," last visited in Jan. 2020. [Online]. Available: https://www.csimn.com/CSI_pages/Modbus101.html

[11] "Message queuing telemetry transport," last visited in Jan. 2020. [Online]. Available: https://mqtt.org/

[12] "Amqp," last visited in Jan. 2020. [Online]. Available: https://www.amqp.org/

[13] "Opc foundation," last visited in Jan. 2020. [Online]. Available: https://opcfoundation.org/

[14] "Distributed component object model (dcom) remote protocol," last visited in Jan. 2020. [Online]. Available: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-dcom/

[15] A. Arsanjani, "Service-oriented modeling and architecture," *IBM developer works*, vol. 1, p. 15, 2004.

[16] O. Givehchi, K. Landsdorf, P. Simoens, and A. W. Colombo, "Interoperability for industrial cyber-physical systems: An approach for legacy systems," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3370–3378, 2017.

[17] E. Tovar and F. Vasques, "Real-time fieldbus communications using profibus networks," *IEEE transactions on Industrial Electronics*, vol. 46, no. 6, pp. 1241–1251, 1999.

[18] "Schema management," last visited in Jan. 2020. [Online]. Available: https://docs.confluent.io/current/schema-registry/index.html

[19] Apache, "apache/avro," Jan 2020, last visited in Jan. 2020. [Online]. Available: https://github.com/apache/avro

[20] "Apache kafka introduction," last visited in Jan. 2020. [Online]. Available: https://kafka.apache.org/intro.html

[21] "Welcome to apache zookeeper™," last visited in Jan. 2020. [Online]. Available: https://zookeeper.apache.org/

[22] "Kafka connect," last visited in Jan. 2020. [Online]. Available: https://docs.confluent.io/current/connect/index.html

[23] "What is elasticsearch," last visited in Jan. 2020. [Online]. Available: https://www.elastic.co/what-is/elasticsearch

[24] "What is kibana," last visited in Jan. 2020. [Online]. Available: https://www.elastic.co/what-is/kibana

[25] "Production-grade container orchestration," last visited in Jan. 2020. [Online]. Available: https://kubernetes.io/

[26] "Run kubernetes everywhere," last visited in Jan. 2020. [Online]. Available: https://rancher.com/

[27] "Eclipse mosquitto," Jan 2018, last visited in Jan. 2020. [Online]. Available: https://mosquitto.org/

[28] "Rabbitmq," last visited in Jan. 2020. [Online]. Available: https://www.rabbitmq.com/

[29] E. Curry, "Message-oriented middleware," *Middleware for communications*, pp. 1–28, 2004.

[30] G. D. S. Ch, C. Venegas, M. Baca, I. Rodríguez, and L. Marrone, "Open middleware proposal for iot focused on industry 4.0," in *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*. IEEE, 2018, pp. 1–6.

[31] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7252.txt

[32] B. Scholten, *The road to integration: A guide to applying the ISA-95 standard in manufacturing*. Isa, 2007.

[33] W. M. Mohammed, B. R. Ferrer, U. Iftikhar, J. L. M. Lastra, and J. H. Simarro, "Supporting a Cloud Platform with Streams of Factory Shop Floor Data in the Context of the Industry 4.0," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, July 2018, pp. 786–791.