International Journal of Engineering Research and Science & Technology

IJERST

ISSN : 2319-5991

www.ijerst.com

Email: editor@ijerst.com or editor.ijerst@gmail.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 8, Issuse.3,July 2020

Industrial Internet of Things Environments: Assessing a Semantic Thing-to-Service Matching Method

L.M.L. NARAYANA REDDY, CHAKKA REDDY USHA

Abstract—

Platforms for the Industrial Internet of Things make it possible to make better decisions based on the data at hand, which in turn boosts efficiency in manufacturing and other commercial proprietary, and coupled with particular IoT gear, data interchange and provisioning between the data sources and platform services continue to be an issue. As a result, we propose and describe in depth an open-source software-based solution called Thing to Service Matching (TSMatch), which enables semantic matching at a fine-grained level between accessible IoT data and services. The report also includes an assessment of the proposed solution's performance in a testbed setting and details its deployment in two distinct Aerospace production scenarios.

Key words

Iot data sources, semantic matching, use cases, and the Iot application are some of the terms that may be found in an index.

INTRODUCTION

The ability to use data available in industrial settings to enhance production and business operations is only one of the many advantages offered by Industrial Internet of Things (IoT) systems. Although data collection has improved, it is still difficult to analyse and use the information gained. In order to function properly, IoT systems need densely populated IoT infrastructures filled with sensors and actuators. Because IoT platforms are often vendor-specific, proprietary, and tied to certain pieces of IoT hardware or cyber-physical systems, putting them up and ensuring they receive proper maintenance may be a challenging task. Carrying the data to the Cloud may be time consuming, error prone, and/or expensive [1] due to the necessity for extra, specialised human intervention. Due to the vendor-based approach, the current IoT ecosystem is fragmented, making interoperability a crucial issue to address, whether from a communication or an application standpoint.

This study is concerned with the second problem and hopes to develop a solution that will facilitate the automatic flow of data between IoT Things (data sources) and the services offered by an IoT platform. To address this problem, we have developed a piece of open-source software called Thing to Service Matching (TSMatch). TSMatch is able to execute semantic matching between services and acquired IoT data at a fine granularity. To that end, TSMatch automates the matching and provisioning process to reduce the complexity of connecting preexisting IoT networks to third-party services. The contributions of this work are I a description of the open-source TSMatch1 engine,

(ii) proof that deploying TSMatch in manufacturing scenarios with realistic operating circumstances is possible, and (iii) an evaluation of TSMatch in a simulated IIoT setting (testbed). The paper will proceed as described below. Following this introductory portion, the associated work is detailed in section II, and the major software components of TSMatch are introduced in section III. Section IV gives a thorough breakdown of the actual implementation. Additionally, section V of the article shows that the system may be easily integrated with other IoT systems as EFPF2. The additional processing time and time to completion components of TSMatch are then assessed in an experimental setting. In section VI, we give the findings and talk about them. Section VII provides a summary and directions for moving forward

Assistant professor1,2
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
P.B.R.VISVODAYA INSTITUTE OF TECHNOLOGY & SCIENCE
S.P.S.R NELLORE DIST, A.P , INDIA , KAVALI-524201

International Journal of Engineering Research and Science & Technology

WORK IN RELATION

There are a variety of methods used in service matching, from the logical [2] to the non-logical [3] semantic based techniques or a hybrid (combining logical and non-logical techniques) approach [4]. For instance, in order to realise a global semantic interoperability solution, Kovacs et al. present the technological approach and the system architecture, which involves merging the FIWARE NGSI and the oneM2M context interfaces. Although this is a potential technique, it is only discussed in theory and not tested in any real-world or experimental settings [5]. Even more, Cassar et al. provide a matching system that incorporates both a semantic and probabilistic matchmaking. The accuracy and Normalized Discounted Cumulative Gain of the suggested method were computed taking into account a dataset of 1007 Web service descriptions in OWL-S. The notion has been validated using simulations, however the suggested strategy shows better performance than previous approaches [6]. We have introduced the approach and the TSMatch idea in past work. Our previous work laid the groundwork for our present effort, which focuses on validating TSMatch in real-world settings once it has been specified and implemented [7].

ARCHITECTURE THAT IS APPROPRIATE FOR THE PRODUCT OR SERVICE BEING PROVIDED

TSMatch is a software platform for semantic matching of Internet of Things (IoT) data to services. TSMatch's primary objective is to automatically deliver data between IoT data sources and services, while meeting the requirements of the services. Presently, TSMatch employs a matching strategy based on semantic similarity to accomplish this goal. The main players in TSMatch are shown on Figure 1. Meaningful representations of the many sensors that make up the Internet of Things are shown here. At now, TSMatch's Engine is a server-based component that may be hosted anywhere from the network's edge to an IoT gateway or even in the cloud. The Thing registry is analogous to a database where details on Things are saved on a regular basis. The TSMatch Client is the programme that the end user downloads and instals on their device, such as an Android phone. The user is synonymous with the recipient of an Internet of Things service, which might be a person, a business, or even a piece of software.

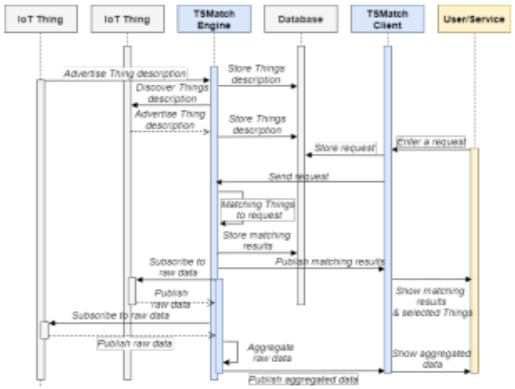

Fig. 1. Sequence diagram of the various actors involved in TSMatch.

As seen in Figure 1, the first step involves the IoT Things broadcasting their descriptions when activated. TSMatch Engine is able to register things in the Things registry and subscribe to events that occur inside them. This is not an internal part of the TSMatch semantic matching engine, but rather is provided by Coaty3 in this instance. The TSMatch Engine may also approach a third party, such a broker, for descriptions of IoT devices. That is to say, it may be set up in the role of a subscriber if necessary. The TSMatch client receives a service request from a user and transmits it to the TSMatch Engine in a separate process. This may be used, for example, to take a temperature reading in a particular room. TSMatch uses similarity matching to determine which IoT Things in the registry best fit the request and the available descriptions in order to provide an appropriate response to the service request. In the event of a successful match, the data gleaned is sent to the TSMatch Client.

The TSMATCH Implementation Considerations

In this section, we'll go through how TSMatch's various parts and interfaces are currently being implemented. All of TSMatch's parts are now dockerized4 since it is built on a micro-service design. The TSMatch engine in this implementation is hosted on an Internet of Things (IoT) gateway, while the client is built for Android. Wi-Fi is used for all of the interaction between the TSMatch components and the rest of the IoT architecture. After that, we'll go through each individual part.

Devices Connected to the Internet of Things

The OGC Sensor Thing API [8] has been used to model the IoT Things. Information on the sensors, including their attributes and whereabouts, is included in the description of the Thing. Coaty, an application-layer communication platform, is used to spread the word. Coaty allows Internet of Things devices to broadcast their semantic descriptions via

multicast, and it alerts subscribers when the device goes offline by broadcasting a "deadvertized" event. In order to find out what's out there, the TSMatch Engine sends out a "discover" event and receives a response with the thing's description. Semantic representations of cyber physical systems using heterogeneous hardware, such as a sensor, a single-board computer (SBC), or a programmable logic controller (PLC) coupled with sensors via its Input/output interface, relate to IoT Things. The sensor driver and a Coati agent are the software components that make up the IoT cyber-physical system and are responsible for handling tasks like publishing information about an IoT Thing.

TSMatch System

Semantic matching between IoT Thing descriptions and service descriptions is implemented on the server side via the TSMatch engine. Requests for services are processed by the engine after being sent by the TSMatch Client. It also manages the matching process between the characteristics and attributes of Things descriptions and the semantic description of services based on queries sent to the TSMatch Thing registry. A Sorensen-dice coefficient and a word frequency-inverse document frequency are used to determine how similar two documents are semantically. Once a matched set has been determined, the accessible Things inside it are gathered together and an aggregated object representing this new set is again persisted in the database. The TSMatch Client receives the matching result and displays it to the user regardless of whether a match was found. If a match is detected, the engine will launch an event that subscribes to data from the chosen sensors and determines an average value for that data set based on the chosen location. The TSMatch Client is then updated with the new information. In the event that a request is removed, TSMatch will discontinue subscribing the TSMatch client to get updates on the IoT Things observations from the broker. Node.js JavaScript and the Typescript programming language5 have been used to create the TSMatch Engine. We have used the string-similarity and natural packages from the Nebulous Plasma Muffin (npm) to implement Srensendice similarity.

Consumer of TSMatch

The TSMatch Client takes care of the service's semantic description, either by constructing one in response to a user's request (such as "monitor temperature") or by retrieving one from a distant location. With the help of the React Native JavaScript framework8, we have created a mobile application called the TSMatch Client. In its present state, the client only supports Android 4.1 and later. TSMatch subscribes to a MQTT broker during launch. Users may browse a catalogue of available Things along with detailed descriptions, articulate their needs, and watch as their data is updated in real time.

Information Management, Search, and Sharing Concerning Things

Coati v2.0 allows IoT Things to register, declare themselves, be discovered, and communicate with the end user (the subscriber). An MQTT broker built on top of Mosquito v2.0.11 facilitates the conversation. The IoT descriptions are kept in a database that uses the JSONB data type for binary storage and retrieval of JSON objects. This database is built on PostgreSQL 13.2. Docker images9 built from the official PostgreSQL source code have been utilised.

Using TSMatch on EFPF Applications

Industry 4.0, the Internet of Things (IoT), artificial intelligence (AI), big data, and digital manufacturing are all represented in the European Connected Factory Platform for Agile Manufacturing (EFPF) ecosystem. The foundation of EFPF is an open, standardised "Data Spine" that allows for the seamless integration of various systems, platforms, tools, and services. The various EFPF parts are supplied by different companies and coordinate their efforts through the system-wide Data Spine. As a result, the EFPF ecosystem is built in a SOA fashion. The EFPF is comprised of the Data Spine, the EFPF Web-based platform (which provides unified access to various tools and services through a Web-based portal), the base digital platforms (four base platforms funded by the European Commission's Horizon 2020 programme), and the external platforms (platforms connected to the EFPF ecosystem that addresses the specific needs of connected smart factories). Currently, TSMatch is one of EFPF's integrated components. Together with the EFPF partner Nextworks10's IoT automation platform Symphony Factory Edition (one of the External Platforms within the EFPF federation), its integration has been tested in production scenarios. Symphony is an end-to-end IoT platform with a flexible design that allows it to work with a broad variety of IoT sensors and actuators, among other types of heterogeneous hardware. Using the EFPF Data Spine, the IoT Symphony platform has been integrated with TSMatch for provisioning and data exchange with the production environments' available IoT data. With input from Walter Otto Muller & Co. KG11 (WOM) and Innovint Aircraft Interior GmbH12, EFPF has integrated and deployed TSMatch in three aerospace manufacturing use-cases (IAI). The applications include: 1) maintaining a constant temperature and humidity in a factory to meet component tolerances (WOM); 2) monitoring raw materials in a freezer to prevent waste from excessive heat (IAI); and 3)

International Journal of Engineering Research and Science & Technology

keeping tabs on a vacuum former from afar to take corrective action as soon as pressure values deviate from acceptable ranges (IAI).

All three scenarios have aimed to protect the consistency and high quality of manufacturing operations by keeping tabs on critical process variables and sounding alerts if certain limits are breached. By using IoT Things, we are able to collect data on the relevant environmental characteristics, which are then sent to the external platform Symphony by means of TSMatch. As can be seen in Figure 2, the Symphony Factory Connector13 makes use of the TSMatch Client to make service requests for IoT Things. TS Match's data is consumed by a Cloud instance of the Symphony IoT automation platform via the EFPF Data Spine, which provides services with interoperable security capabilities. The Symphony HAL (a software module that primarily abstracts the low-level details of various heterogeneous fieldbus technologies and provides a common interface to its users), Symphony Data Storage, and Symphony Visualization provide the visual monitoring, sensor data and event storage, signal analysis, and alarm systems for the Things. Actions, such as control actions on field-level devices, notifications (emails, SMS), and alarms, are determined by the Symphony Event Reactor after merging data from various sources and data brokers (e.g., TSMatch) (via stack light which provides visual and audible indications). In the Cloud deployment of the platform, the real-time data and the current state of the thresholds and alerts may be seen and managed using the Symphony GUI.

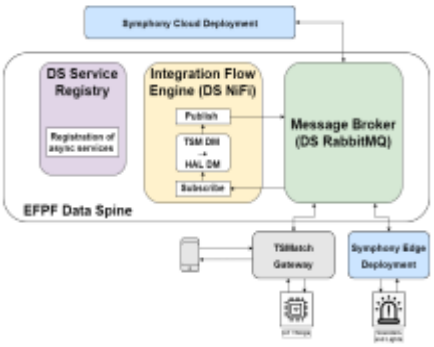

Fig. 2. EFPF interconnected components on the manufacturing use-cases.

As illustrated in Figure 3, a successful deployment has been done on production sites. The EFPF components, including TSMatch, have therefore been validated based on the defined requirements, usability aspects, as well as ease of installation/configuration. After 6 months of deployment, traceability and detection of abnormalities has been also achieved, which increased the reliability of the manufacturing process, reduced delivery delays, and minimized rejects/waste occurrences. Fig. 3. Installation of the use-cases.

PERFORMANCE EVALUATION AND RESULTS

TSMatch Testbed

Fig. 3. The TSMatch demonstrator at the forties IIoT Lab.

We have established a TSMatch testbed on the forties IIoT Lab14 based on the operational requirements generated from the TSMatch integration on real-world industrial use-cases, as shown in Fig. 4. As shown in Figure 3, the testbed consists of the following parts:

Two real Internet of Things (IoT) devices and ten simulated IoT devices are available, with the former sporting a total of five sensors (for measuring things like temperature, humidity, sound, air quality, and particles in the air). According to section IV, each virtual IoT thing is linked to a virtual IoT sensor, each of which is housed in its own Docker container.

• TSMatch Engine, Message Broker, and Database: TS Match Engine is containerized, and the Mosquito message bus and PostgreSQL database are also deployed as containers in the forties IoT gateway.

Using a MQTT client15, the IoT service request simulator may mimic third-party IoT services. TSMatch Client was abandoned in favour of the service request, which allowed for precise regulation of the inter-request time gap.

International Journal of Engineering Research and Science & Technology

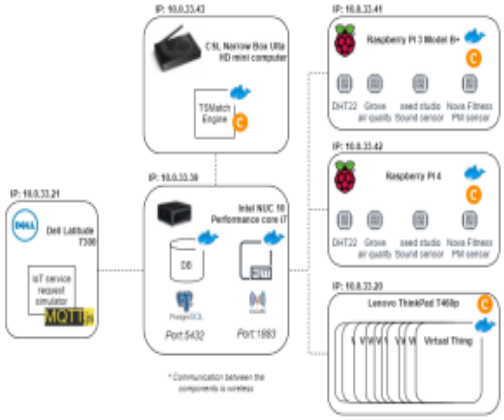

Fig. 4. TSMatch components and experimental interconnections.

The hardware specifications for each of the devices that is part of the demonstrator are given in Table I.

conclusion

TSMatch is a new approach described in this study for semantically matching IoT data sources and service descriptions. We outline the TSMatch software architecture and discuss its practical applications in the industrial.

TABLE I MEAN PT AND TTC OF SEQUENTIAL AND SIMULTANEOUS SCENARIOS.

	Sequential Scenario						Simulta-neous Sce-nario
	Fixed TI			Variable TI			
	Low Range	*Medium Range*	*High Range*	*Low Range*	*Medium Range*	*High Range*	
Mean PT(ms)	151.88	32.76	33.72	55.8	53.76	72.84	74.8
Mean TTC(ms)	321.56	170	111.08	127.8	198.88	260.96	237.04

contexts of the European EFPF project settings. The report also includes a first performance assessment of TSMatch and details the existing open-source TSMatch implementation's processing time and time to completion of requests. In order to identify the "most" appropriate collection of IoT Things whose aggregated data may satisfy a certain semantic request, future work will concentrate on enhancing the semantic matching engine by including a more intelligent parsing and also learning.

REFERENCES

[1] A. Verma, and S. Kaushal," Cloud computing security issues and challenges: a survey," In International Conference on Advances in Computing and Communications Springer, Berlin, pp. 445-454, July. 2011.

[2] A. Segev and E. Toch, "Context-Based Matching and Ranking of Web Services for Composition," in IEEE Transactions on Services Computing, vol. 2, no. 3, pp. 210-222, July-Sept. 2009.

[3] H. Fethallah, A. Chikh , and A. Belabed. "Automated discovery of web services: an interface matching approach based on similarity measure." In Proceedings of the 1st International Conference on Intelligent Semantic Web-Services and Applications, pp. 1-4, 2010.

[4] M. Klusch and K. Patrick, "isem: Approximated reasoning for adaptive hybrid selection of semantic services," In Extended Semantic Web Conference, Springer, Berlin, Heidelberg pp. 30-44, 2010 .

[5] E. Kovacs, M. Bauer, J. Kim, J. Yun, F. Le Gall and M. Zhao, "Standards-Based Worldwide Semantic Interoperability for IoT," in IEEE Communications Magazine, vol. 54, no. 12, pp. 40-46, December 2016.

[6] G. Cassar, P. Barnaghi, W. Wang and K. Moessner, "A Hybrid Semantic Matchmaker for IoT Services," 2012 IEEE International Conference on Green Computing and Communications, pp. 210-216, 2012.

[7] N. Bnouhanna, R. C. Sofia, and A. Pretschner,, "IoT Thing To Service Semantic Matching," 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), pp. 418-419, March 2021.

[8] S. Liang, C.Y. Huang, and T. Khalafbeigi. "OGC SensorThings API Part 1: Sensing, Version 1.0.", 2016.

[9] I. Martens, D9.1 - Implementation and Validation through Pilot-1. [Deliverable] https://www.efpf.org/deliverables, 2021.

[10] I. Martens, D9.2 - Implementation and Validation through Pilot-2. [Deliverable] https://www.efpf.org/deliverables, 2021.

[11] I. Martens, D9.3 - Implementation and Validation through Pilot-3. [Deliverable] https://www.efpf.org/deliverables, 2021.