Scalable approaches to integration in heterogeneous IoT and M2M scenarios

Alex C. Olivieri, Gianluca Rizzo
Institute of Information Systems, HES-SO Valais, Sierre, Switzerland
alex.olivieri@hevs.ch, gianluca.rizzo@hevs.ch

***Abstract*—The Internet of Things (IoT) opens new perspectives for the Machine-To-Machine communications, as it brings to settings with a large and heterogeneous set of devices. This makes integration of such diverse technologies a challenging task. A typical solution to this problem is represented by *universal gateways*, which provide internal semantics for protocol translation. However such approach is bound to be challenged in future IoT scenarios, as it brings substantial performance impairments in settings with very large number of devices and of technologies. We address these limitations, by proposing two novel approaches. A first is based on distributing the management of the different technologies among gateways. A second approach makes use of Web Service delegation, with gateways acting only as a connecting point between the entities and some service that can interpret the information exchanged. We implement and validate experimentally these approaches, showing that they both scale sensibly better than the traditional approach, guaranteeing acceptable performance even with a high degree of heterogeneity. Moreover, we identify the criteria which should be considered when choosing between the two proposed approaches. Our results establish a set of guidelines for integration in large and heterogeneous IoT scenarios.**

***Keywords*-Internet of Things; machine-to-machine.**

I. Introduction

Machine-To-Machine (M2M) interaction is an hot topic in the Internet of Things (IoT) domain, and it refers to the information exchange among devices without human assistance. In the beginning, M2M was used to create pretty simple systems that were task or device specific, and where only one technology was employed. This, due to the absence of a standardized M2M platforms that would allow the interconnection of heterogeneous technologies [1]. These M2M systems are pretty simple because they must address only one technology at a time. They need only an *interpreter*, which translates data to and from the couples of communicating devices.

The set of services and functionalities needed to enable such interaction are generally present only in more recent technologies, but they are totally absent in legacy, non-IP technologies such as in old Building Automation protocols. This is an important issue, because these devices often called *legacy devices* still remain the majority of the *things* involved in the IoT paradigm.

With the gradual diffusion of the Internet of Things, which enable interactin among a large and very diverse set of devices, the M2M paradigm is poised to evolve from the usual mono-technology scenarios to scenarios involving several different technologies.

To address the issue of making these legacy devices interoperable not only within their protocol domain, but in a multi-protocol way, the usual solution is the use of *universal multi-protocol gateways* as a system that groups many different technologies under an internal semantics [2]. These gateways solve the protocol translation issue, but they encounter two limitations that in the IoT can lead to really inefficient implementations. The first concerns the high engineering costs in managing a high number of technologies in an efficient manner. It can be seen as the incremental costs of adding a new technology to a given scenario. The second problem is more strictly related to scalability. Indeed, in many IoT scenarios it would be unfeasible to have a single gateway managing all the devices, as it could generate a bottleneck effect on system prformance. In a context in which devices grow in number constantly as they get increasingly powerful, small, cheap and therefore ubiquitous, and in dynamic settings where fixed and mobile devices interact, adopting solutions which scale efficiently is essential for an acceptable quality of the services delivered by such IoT infrastructure.

This paper proposes two distributed approaches, respectively called *distributed gateway* and *web service gateway*, in order to address the issues left open by the universal gateway approach. The Distributed Gateway approach consists in partitioning the functionalities required for protocol translations, and in distributing them among a set of gateways which coordinate and interact. With this approach, each gateway typically supports only a small set of protocols, relying on other gateways for all the other protocols. The web service gateway approach instead delegates all the workload needed to interpret the data coming from devices to web services, which are accessed on demand. Under this approach, gateways are typically quite simple, acting only as a data forwarder from devices to the web service and vice-versa.

In order to evaluate these approaches, we have implemented and tested them experimentally, checking their scalability and feasibility. Our results draw some guidelines that the designers and developers of such gateways can follow to create efficient and scalable IoT integration solutions.

The paper is organized as follows. In Section II we provide some background on Machine-To-Machine interactions in the Internet of Things, and we discuss the main issues arising

from traditional approaches. In Section III we present our two approaches, and in Section IV we describe how we have modeled and implemented them on an experimental scenario. In Section V we describe our experimental set-up and our tests, commenting on the results. Finally, Section VI concludes the paper.

II. Machine-to-Machine interactions: adaptation to the Internet of Things

In this section, we explain the changes that IoT brings to the M2M interactions and why these changes make the usual management of these interactions inadequate.

A. Evolution of the Machine-to-Machine paradigm

M2M has to evolve to face the new challenges brought by the Internet of Things and, more generally, by the Internet. Nowadays, M2M involves communications between machines from different vendors, typically using different technological communication protocols.

The systems must evolve, proposing new solutions to face the heterogeneity that will be present in those scenarios. The solutions that emerged are often referred as Universal Gateways [3]. A Universal Gateway is a device that transacts data between two or more data sources using communication protocols specific to each one.

B. Traditional M2M Gateway Approach

The history of M2M interactions showed over time the utilization of a local scenario M2M Gateway, where only a few solutions contemplated the integration of heterogeneous protocol and only a limited number of devices were used in order to perform a bounded task or group of tasks [7]. This solution was efficient and successful until the Building Automation System did not advance to the current situation, and the Internet of Things did not create such scenarios where huge number of devices may be employed. Figure 1 shows the approach Multi-Protocol Gateways usually use. The Gateway can hypothetically manage, if equipped by developers of the necessary structures, all protocols for building automation technologies. A Gateway is implemented to fully support all devices belonging to different technologies, which are employed in the scenario of interest, and only one Gateway is needed to manage that scenario. From our point of view, as the M2M interactions domain is changing, this way of developing Gateways can encounter two developing issues. **Heterogeneity problem** as difficulty in creating solutions that can face the high and mutable number of heterogeneous technologies of the devices employed in current scenarios. **Scalability problem** as difficulty in creating solutions which scale well with the increasing number of devices.

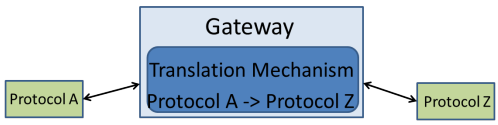

Figure 1. Multi-Protocol Gateway

1) Heterogeneity Problem: Nowadays, Internet of Things scenarios expect the usage of devices belonging to various heterogeneous technologies, they also expect that those devices can be replaced over time with devices belonging to other technologies. It means that developers of a Gateway should have technical knowledge of, hypothetically, all possible technologies for devices within the Internet of Things context. To have such knowledge of a large number of technological protocols can be really challenging. Each technology possesses features that are different and all of them use different protocol stacks. To create and maintain such an idea of a gateway can create a huge effort for the gateways owners. The team that implements it, having first identified the interesting technologies, must foresee how many different developers it needs and must anticipate specialization courses in order to give them the necessary knowledge.

2) Scalability Problem: The Internet of Things brought Gateways to the next level, where it is difficult to estimate the number of devices that will be employed on a scenario from the beginning. It means that the Gateway can start working perfectly with a certain number of already tested devices, and they can deal with the simultaneous incoming/outgoing communications from and to devices properly. The problems arise when the number of devices increases with likely consequent augmentation of the number of simultaneous communications. If a gateway was modeled without load balancing features, it is likely that it would collapse under the new workloads the Internet of Things scenarios can produce.

III. Novel M2M Gateway Approaches

In this section we propose two approaches that can enrich the current M2M Multi-Protocol Gateway solutions by addressing the problems described in II-B1 and II-B2. Both approaches aim to distribute the devices management in order to ease the development and to provide scalability to current scenarios.

A. Distributed Multi-Protocol

Figure 2 shows the first proposed innovative approach. In this approach, we propose to treat the protocols singularly or as a subset of the selected protocols in each gateway. The devices managed by each gateway belong to a unique technology (or in some case to a small subset of them). Each gateway is capable of interpreting the information contained in the raw data exchanged with the devices attached to it. Afterwards, gateways provided by different working groups skilled in dedicated technologies can be combined to create multi-protocol scenarios. At first sight, this approach of

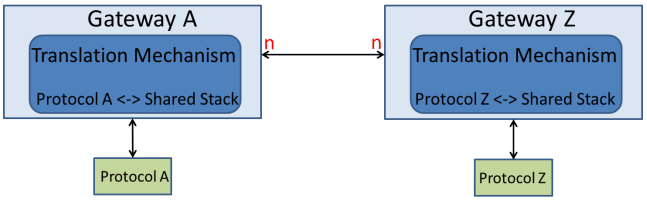

Figure 2. Distributed Multi-Gateway

having every Gateway managing only one protocol may seem like the multi-protocol Gateway is regressing back some years ago. However, we believe that a proficient engineering of the systems will bring advantages which will eliminate the drawbacks if this approach is used. By using this approach, it is possible to envisage a solution for the two challenges previously illustrated as explained as follows. **Heterogeneity Solution:** Developers of a Gateway must have knowledge about only one technology to be able to map the data coming from devices to the internal semantics of the gateway and vice versa (from the semantic to the data for the devices). This allows developers to enrich their competences for a smaller domain of technologies that will lead to an improvement of the quality of their work. Obviously Gateways that manage different technology can be employed to create heterogeneous scenarios, but each of those gateways will be developed by different vendors. **Scalability Solution:** This approach faces the scalability issues deriving by the growth of the devices used in novel scenarios distributing the managing of devices among more distributed Gateways. By doing so, we can alleviate the scalability constraints arising from the underlying framework present inside the gateways. In fact, developers can try to make a framework as scalable as they can, but eventually the scalability depends upon the hardware and the number of processes and threads the underlying platforms can support. So, the only way to improve the scalability is to distribute the amount of work among more Gateways. Nevertheless, this approach also brings some new challenges which the developers must face. Now, every Gateway must be able to interact with other Gateways to implement scenarios. We are not specifying which kinds of communications will be used, because it depends on the design of the Distributed Multi-Protocol System. But to interact, the Gateways need to talk the same language, which leads to the creation of a common language spoken by all gateways.

B. Web-Service Multi-Protocol

Figure 3 shows the second proposed innovative approach. This approach deals with the heterogeneity of the protocols with a methodology similar to the ones we proposed for the Distributed approach, where only one protocol or a subset of the selected protocol is managed within a single Gateway. As before, Gateways managing dedicated technologies must be combined to create multi-protocol scenarios. This main difference is that in this approach, a solution where the Gateways are fully unaware of the format of the data they will exchange with the connected devices is proposed. It receives raw data from devices and sends raw data to devices without being able to interpret the information contained in the raw data. To have interpretable data, the gateways need to contact a Web Service that can perform the translation between raw data and shared format that they will use for internal purpose and to interact with other Gateways (and to translate the information from the shared format to the raw data). The Web Service performs the task of interpreting the raw data belonging to different protocols and translates them into a shared format. This task can be accomplished thanks to so called Translation Modules that developers, experts in some technologies, implement and install on the Web Service. We believe that this approach brings solutions for the two challenges previously mentioned in the following ways: **Heterogeneity Solution:** Developers who build a gateway do not need any knowledge of the raw data exchanged with the devices, because the interpretation work will be performed in the Web Service, through the Translation Mechanisms. This allows developers to focus on the logic of the Gateway and how it manages the incoming/outgoing communications with the devices. **Scalability Solution:** The translation needs computations regardless of the methodologies used to perform it. Those computations could lead to some performance decreases when the number of devices connected grows too much. This decrease in performance does not happen in Web Service, because they can be implemented in a way that can afford higher workloads. With this approach, it is possible to let the Web Service work for the Gateway, while the latter will focus on different tasks thank to the possibility to have asynchronous communications [4] between the Gateways and the Web Service.

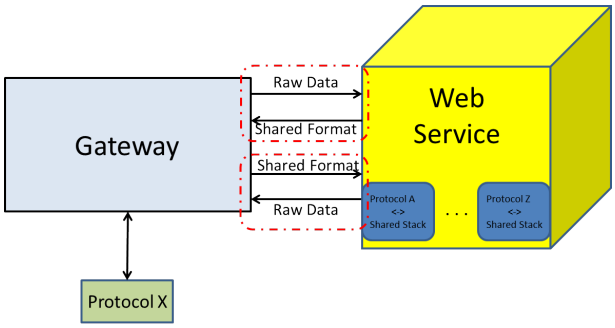

Figure 3. Web-Service Multi-Protocol

IV. MODELS

In this Section, we explain how we modeled the three M2M interaction systems approach (Traditional, Distributed,

Web-Service). The idea is to have each system composed of the same components in order to reproduce homogeneous conditions when the tests are performed. What changes is how those components are distributed.

A. Components

We stated that when M2M interactions are applied in the context of the Internet of Things, the former gateways are no longer satisfactory. This is due to the heterogeneity of the devices involved and the complexity of the scenarios.

A gateway is a system that exchanges data with the devices attached to it and makes available those data to other components. In fact, gateways are usually parts of complex scenarios, where those scenarios need to interact with the devices it manages. For the purpose of this paper we do not consider the logic of the scenario that can be managed by Decision Makers or Control & Monitoring Systems, because it is out of the scope, but we focus only on the data exchange and translation part.

In our models we envisaged two main components to be implemented and deployed - the Connector and the Translator.

1) Connector: The Connector within the context of this work is considered as the component to which all devices are attached and acts as a communication bridge between the devices and the Translator. It receives measurements from devices that act as sensors, forwarding them to the Translator and sending data to devices that act as actuators. Hence, it is the manager of the information workflows in the form of raw data.

When a device is attached to the gateway, the gateway assigns a global unique identifier to that device based on the IPv6 Mapping Module [5]. This mapping module allows every gateway to identify the technology to which a device belongs, and it is a very important aspect for the data interpretation phase. Within this global unique identifier, we can find other information useful for applying the distributed idea proposed in the Distributed and Web Service approaches. The three important fields of that identifier are Gateway Identifier, Protocol Identifier and Device Identifier. With this identification mechanism, we can aggregate more gateways in distributed scenarios while still having the capability to understand which devices are being referred to.

2) Translator: The Translator is the component that can interpret information contained within raw data sent by devices and can translate it to a defined format (in our case we use the JSON format) understandable by other components. Vice-versa, it has the ability to translate information formatted using JSON into raw data. To perform these operations, the Translator must know the technology to which the device that is sending the data belongs, and this piece of information is passed by the gateway using the global identifier of the device, along with the raw data.

B. Interactions

In this subsection we explain how the components are deployed in order to implement the different approaches. We also describe the communication methodologies used between the Devices and the Connector and between the Connector and the Translator. Changes of these methodologies would affect the performance.

1) Traditional M2M System and Distributed M2M System: The traditional approach and the distributed approach have been grouped because from a model point of you they are identical. The difference between them is the number of technologies they manages. The deployment reflects the centric-approach used by usual Multi-Protocol Gateway, where the Connector and the Translator are placed into a single Workstation.

- Devices - Connector: devices are implemented as software modules that simulate measurements. They send raw data to the gateway calling a method it provides as entry point. (The interaction is a method call.)
- Connector - Translator: when a gateway wants to translate information from raw data to JSON (and vice versa), a thread is started and it passes the information to that thread. Connector and Translator are in the same memory space. (The interaction is a synchronous request to a local thread.)

2) Web-Service M2M System: Here, in addition to the Workstations acting as Connector, a Web Service is added to the system in order to translate the information from raw data to JSON (and vice versa).

- Devices - Connector: devices are implemented as software modules that simulate measurements. They send raw data to the connector calling a method it provides as entry point. (The interaction is a method call.)
- Connector Web Service (Translator): when the connector wants to translate information from raw data to JSON (and vice versa) it calls an HTTP method request in the Web Service and passes the information as request parameter. Connector and Translator are in different locations. (The interaction is a standard HTTP request.)

V. Tests

In this section, we test the scalability and the feasibility of the three approaches. At the beginning we clarify the objectives of the tests and we define the metrics used for the tests. Then we compare the results of the tests in order to evaluate them and to design development guidelines for the gateway's implementation and deployment.

The focus of the test is: a) to test how the different systems responds to the variation of the workload given by the changes on the number of contemporaneous communications to which a system has to deal with; b) to test the difficulties

the developers find in implementing the different approaches the systems are based on.

As said in section IV-B1 the traditional and the distributed approaches are equivalent from the scalability point of view, since both incorporate the Connector and the Translator on the same workstation. For this reason we perform a single test for these approaches and another test for the web-service approach.

A. Set Up

In order to implement our models, we made use of workstations and web-services. Workstations had a Intel Xeon CPU E31245 3.30 GHz processor, with 16GB of RAM and a 64-bit operating system. Web service platforms have been implemented over Amazon Cloud, on a Tomcat platform with load balancing, and auto scaling.

B. Test Metrics

In this subsection, the metrics used for the tests are defined. Two features are to be measured: feasibility of the systems implementation and the scalability of the systems.

Scalability: We measure the interval of time between the instant when a device sends raw data to the Connector, and the instant when the Translator provides to the Connector the same piece of informations in JSON format.

Feasibility: We check the feedback provided by developers to a questionnaire that contains closed questions about the difficulties of implementing the different systems.

1) Scalability: Each test was performed 30 times and the reported values represent the average of the measurements. For the Translation test, the number of devices attached to the Gateway that send measurements at the same time are considered and the results represent the time needed to provide the translation for all measurements. The samples taken for the tests go from 10 devices sending measurements to 10,000 devices, all sending measurements at the same time. Figure 4 is related to the translation performed

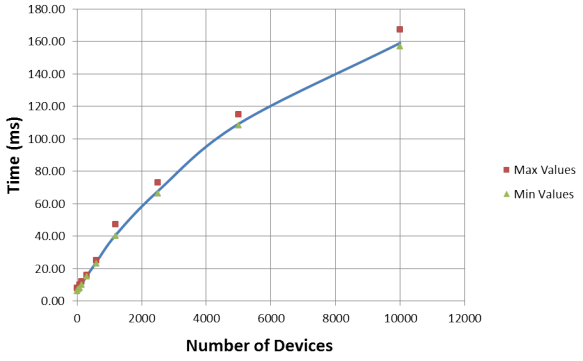

Figure 4. Latency when the translation is performed within the workstation

within the workstation (Traditional approach and Distributed approach). It shows the curve that represents the relation between the number of devices that send information at the same time and the interval of time the Translator included within the Gateway needs to provide all information translated. It can be seen that the latency is almost zero when only a few devices send information and it increases when the number of devices grows, following a slightly logarithmic increasing trajectory. Another observation is that the variance in the measurements in each sample is very limited.

Figure 5 shows the curve that represents the relation between the number of devices that send information at the same time and the period of time Gateway needs to obtain all information translated when the translation is demanded from a Web Service. Unlike what Figure 4 shows, this curve also presents latency in the situation when there are just a few devices that send information. In this test, the increment when the number of devices grows has a linear trajectory. We also notice that there is a strong variance in the measurements in each sample, which we believe is provided by variance in the transferring of data via the Internet.

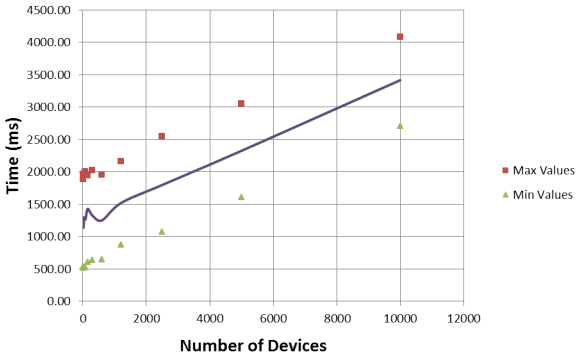

Figure 5. Latency when the translation is demanded to the Web Service

2) Feasibility: To evaluate the Feasibility, or as we defined before the Easiness of creating the different systems, a test was submitted to all of the developers that worked on the systems prototypes. The Developers had to answer with a number ranging from 0 (no effort) and 5 (very difficult). The test contained four questions that we believe are fundamental to evaluate the difficulties developers can find when they decide to implement systems. After the developers answered the questions, an oral session was held in order to understand the reasons for their answers and what obstacles were more challenging to surmount. In the project design and implementation, we included Bachelor students who in the beginning of the work, had existing knowledge of language programming, but did not have knowledge of Building Automation Systems, Distributed Programming nor Web Service. This was done to ensure the coherence of the tests. Figure 6 shows the answers provided by the

	Developer 1	Developer 2	Developer 3	Developer 4
How difficult was learning a Building Automation Technology?	4	5	3	5
Once a technology is assimilated, how difficult is to manage new devices belonging to it?	2	3	2	2
How difficult was learning how to implement in Distributed Environment?	3	3	3	3
How difficult was learning how to implement using Web Services?	2	3	2	3

Figure 6. Questionnaire about Easiness

developers and shows that to learn new Building Automation Technology was the most challenging task they dealt with. However, when a technology is assimilated, it becomes easier to learn how to manage new devices belonging to that technology. The Distributed Programming had some difficult issues in the beginning, but considering the amount of online documentations, tutorials and books about the topic, they said that they could understand quite quickly how it works and how to implement it.

C. Evaluation

In this section, the results of the tests performed are evaluated and a choice of the recommended approach is suggested for implementing the Gateways, with the corresponding motivation.

1) Scalability: In Section V-B1, we tested how two different approaches scale when the number of translation requests increments. The trajectory showed in figure 4 is logarithmic suggests that a gateway can react well to the increments of the number of simultaneous requests. And by comparing the results showed in figure 4 and 5 we can claim that there is no need to delegate the translation to a web-service for scalability reasons.

It is a decision of the system's designer to decide how many devices associate to a gateway depending on the defined acceptable latency.

2) Feasibility: The evaluation of the feasibility depends upon the answers the developers provided during the questionnaire and the subsequent interview regarding their feedback. The developers expressed difficulty in learning how Building Automations Systems functions, due to each having different features and that it was based on different stacks that make it even more difficult to master properly. The main reason for their concerns is due to a lack of how to do tutorials, which allows developers to fully study the protocols in order to understand how they can be managed. During the implementation, the developers tried to use third party software libraries that already managed the protocols and making it easier to integrate various technologies together. Regarding the Distributed Programming, the developers found difficulties at the beginning of the work, but once they understood the paradigms and patterns on which it is based, the work became much easier.

VI. Conclusions

The following conclusions have been drawn after analyzing the test results based on the defined parameters for the evaluations:

a: To group all Building Automation Technologies, or a big subset of them, under a unique Multi-Protocol Gateway can be very challenging and requires developers to invest a lot of time and effort to master them properly.

b: The M2M Interaction Systems we have tested demonstrate that the translation from raw data into a shared Semantics does not imply scalability issues. The reason is that the task is so fast that the amount of measurements that could arrive at the same time can be well managed also if high number of requests arrive at the same time.

As guideline to developers we suggest to create gateways mono-technology or that manage a very low number of technologies and distribute the technologies among more gateways. For the gateways we suggest to use the maximum they can API library and gateways already existing. Their main focus should be on a semantic layer to allow the different gateways to interact and cooperate.

We believe that following these guidelines in this deliverable, the management of more sophisticated scenarios can be allowed, such as complex real-time scenarios, which currently are considered very challenging in the Internet of Things domain.

References

[1] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. Johnson, "M2m: From mobile to embedded internet," *Communications Magazine, IEEE*, vol. 49, no. 4, pp. 36–43, April 2011.

[2] M. Jung, J. Weidinger, C. Reinisch, W. Kastner, C. Crettaz, A. Olivieri, and Y. Bocchi, "A transparent ipv6 multi-protocol gateway to integrate building automation systems in the internet of things," in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, Nov 2012, pp. 225–233.

[3] J. Latvakoski, M. B. Alaya, H. Ganem, B. Jubeh, A. Iivari, J. Leguay, J. M. Bosch, and N. Granqvist, "Towards horizontal architecture for autonomic m2m service networks," *Future Internet*, vol. 6, no. 2, pp. 261–301, 2014. [Online]. Available: http://www.mdpi.com/1999-5903/6/2/261

[4] E. Johnsen and O. Owe, "An asynchronous communication model for distributed concurrent objects," *Software & Systems Modeling*, vol. 6, no. 1, pp. 39–58, 2007. [Online]. Available: http://dx.doi.org/10.1007/s10270-006-0011-2

[5] R. e. a. G, "IPv6 mapping to non-IP protocols," Internet Requests for Comments, RFC Editor, RFC draft, September 2014. [Online]. Available: https://tools.ietf.org/html/draft-rizzo-6lo-6legacy-02