An Self-configuration Architecture for Web-API of Internet of Things

Eric Bernardes Chagas Barros and Admilson de Ribamar L. Ribeiro
Computing Department, Universidade Federal de Sergipe, Aracaju, Brazil

Keywords: IoT, Web-APIs, Self-configurable, Internet of Things, Web of Things, WoT, Self-configuration, REST, Mote.

Abstract: The internet of things (IoT) is the paradigm that will dominate the computing world in the coming years. In this way, studies should be conducted in such way to ensure its enhancement and in the quest for that improvement is necessary to use the already existing technologies that apply to IoT. This paper's purpose is to unite different technologies like REST, cloud computing and embedded operating system in order to obtain mechanisms capable of self-configuration. Thus, it was possible to conclude that the architecture proposed would increase useful techniques for the implementation of systems that want to run the self-configuration as well as assist in setting up networks of computers that work with wireless sensors and IoT.

1 INTRODUCTION

Internet of Things (IoT) is the paradigm that until 2025 will dominate the world of computing (Atzori, 2010). The ubiquity of the Internet in less than 20 years almost connected all people in the world and has generated new demands for space. Now people not only need to exchange information and services, but also objects. Although for this to occur it is still necessary that both, the technology and society are prepared.

The Things which refers to this paradigm are related to devices or motes that are arranged in an environment and have their own characteristics, how to measure a temperature, check if the light is on, if a window is open, the amount of milk in a refrigerator, among many other possibilities. By capturing the information for which they were programmed, these devices send data through existing Internet's services to let user become aware that these data are stored and can provide some useful information to him. The use of existing resources on the web by these motes can be done through APIs and can be characterized by the use of cloud computing, when this happens it is customary to call these APIs of Web-APIs (Zeng, 2011).

The configuration and installation of devices that will integrate a large and complex systems within the IoT is a challenge that is time consuming and error prone, even for the great specialists (Kephart, 2003). Moreover, the large growth of network nodes made with different technologies and different platforms can result in a hard and repetitive work.

Therefore, to facilitate the use of such devices, and applications' development, the use of techniques that enables the system to adapt itself to the environment and self-configure is a great need. However, the Web-APIs that exists in nowadays has not yet incorporated these concepts yet.

This paper aims to introduce a mechanism of self-configuration for the Internet of Things, where the main idea is to make easier the configuration of devices and Web-APIs that will control the environment.

This paper aims to introduce a mechanism of self-configuration for the Internet of Things, where the main idea is to make easier the configuration of devices and Web-APIs that will control the environment.

The remainder of this paper is organized as follows. In section 2, will be shown on the existing requirements nowadays and that contribute to the development of Web-APIs. Section 3 talks about the Web-APIs that exists in the market and the summary of its main characteristics. Finally, in Section 4, the proposed architecture will be explained and a possible mechanism architecture that can be used in the development of self-configurable Web APIs. Conclusion and future research hints are given in Section 5.

328 Bernardes Chagas Barros E. and de Ribamar L. Ribeiro A..
An Self-configuration Architecture for Web-API of Internet of Things.
DOI: 10.5220/0004944003280334
In *Proceedings of the 10th International Conference on Web Information Systems and Technologies* (WEBIST-2014), pages 328-334
ISBN: 978-989-758-024-6
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)

2 REQUIREMENTS OF WEB-APIS IN INTERNET OF THINGS

Currently, the existing Web-APIs have a set of basic characteristics that are used to carry out the communication of devices with the Internet or for needs of these motes due to its limitations. These fundamental features are described in this article in order to enumerate some of the concepts that can be used to serve as basis for self configuration mechanism, such as the form of communication with Rest (Zeng, 2011), storage and standardization communication through the use of markup data languages (XML, YAML, JSON) (Xively 2013).

2.1 Open-source

Although this characteristic is not a specific functionality that help directly the devices, it was regarded as important for that in the future people will work on top of existing Web-APIs and make your code to be improved and become Customer self-configurable.

This term refers to the so-called free software, where to be held a consolidated distribution, is also distributed its source code for that can be freely used, modified and shared by its users.

2.2 Rest

The REST-based architecture is considered "the true architecture of the Web" (Zeng, 2011), it is based on the concept that everything is modeled as resource using the HTTP URI. Thus, customers can identify the resources they need through the URI, manipulating them through traditional HTTP commands like: PUT, GET, POST and DELETE. The PUT and DELETE.

Moreover, it has self-descriptive messages, i.e, the resources are free to make their own representations of data format. Obviously, end-systems must agree with this representation so that communication can take place properly. In this way, it is possible to use HTML, XML, text, PDF and images as the format of data to be sent.

Another important feature is that REST works with stateless requests, treating each request independently, and this may not require a server to store session information or the status as is each of the multiple acquisitions. However, statefull interactions can be supported in REST through the use of hyperlinks, so the states of the resources can be transferred by means of URIs for cookies or hidden fields (Zeng, 2011).

2.3 Standardization

As the APIs and the devices are usually developed in different languages, it must be pre-established a format of data communication between the receiver and transmitter and how they will exchange messages to inform how the data is separated and what the content within it represent. Consistently, to earn this type of representation the IoT sought markup languages known data, such as XML, JSON, YAML or CSV.

These languages are very portable because it does not depend on hardware or software platforms to work and any databases can communicate with each other through them. By having the ability to self define data, as well as having the characteristics described above, these languages are used for interoperable networks, allowing objects of different characteristics understand each other.

2.4 Centralized Architecture

Due to the limitations of the devices many of the activities more robust need to be sent to a server that has capacity to perform a greater load of processing and storage. Therefore currently the Web-APIs, tend to be centered on a server that is able perform this type of activity. Thus, a network IoT using these Web-APIs tend to use the REST to communicate with a server that is receiving data and managing the devices in the network.

2.5 Security

When the term security is mentioned, the first word illustrated is identification. In IoT, recognition of each device with the use of traditional IPs. Despite this, only a network identification is not sufficient to ensure the safety, it is necessary a profile control to inform if this equipment has access to the service that it is requesting. As in IoT these services are provided by APIs, the controls of inflows are usually made by API-Keys.

Within the API-Key are encapsulated three types of permissions that operate in a hierarchical manner: object key (the general key of the API), object permissions and the permissions of features of objects, the latter being optional. The general permissions objects keys are created for your applications to have access to APIs. Each application may ask how many objects keys you need.

An object key can have multiple objects permissions (it is mandatory at least one) and each acquiescence of object contains a set of different permissions. For example, a key can be created to allow a read-only access to the entire public resource available, in the same way, can allow a write access to a resource responsible for supply of data by means of a specific IP (Xively, 2013). There are still the feature permissions that serve to restrict access to a given resource. These permissions, as were elucidated above, are optional.

In addition to the API-Keys which grants the security level of access control, there is the HTTPS that provides security at the level of sending and receiving data throughout an encrypted and secured channel. As the APIs' principal way of communication focus on HTTP (REST and SOAP) the use of HTTPS helps to prevent the attacks of type man-in-the-middle, seen that the HTTPS is the implementation of HTTP on top of the SSL/TLS protocol to provide authentication of hosts purposes and encrypted communication between them.

2.6 Self-configuration

With the advance of network technologies, devices shipped and software tools, the growth of heterogeneous nodes, of great complexity and extremely dynamic that pass to operate within the Internet becomes something that cannot be easily administered.

The autonomic computing is inspired in the human being's nervous system. Its main objective is to develop applications that can self-according to guidelines imposed by human beings at a high level. Thus with the policies established at a high level it is possible to make the systems self-reliant to self-configure, self-healing, self-optimization and the self-protection (Kephart, 2003).

The self-configuration is responsible for automated configuration of system components, with it the system will automatically adjust and it always will adjust based on policies of self-configuration. The self-optimizing components and systems continually seeking opportunities to improve their own performance and efficiency. Self-healing the system automatically detects, diagnoses and repairs problems of software and hardware located. The Self-Protection system automatically if defends against malicious attacks or cascading failures. It uses earlier warnings to anticipate and prevent failures in the entire system (Kephart, 2003).

2.7 Code-source Device

As each device needs to communicate with the Web-API through the REST, many of these offer codes-sources for which the user copy and paste in Integrated Development Environment (IDE) responsible for programming the device. In this way it is possible to at least have an example that how to program a device and if it is possible already find a code that is applicable to the device that will enter the network.

However it is important to realize that although there may be a useful source code available for the used equipment perform the copy and paste codes for multiple appliances can be an arduous task and subject to errors even for the great specialists, once that may exist dozens of these to be configured in a single environment.

2.8 Storage

The storage of data in IoT is an interesting area of study, once the majority of devices that exist in a network of this type do not have large storage capacity. In contrast, the data used in the communication are stored in a central device that has the characteristics necessary for the recording of relevant data to the system.

As the WEB-APIs are on a server that contains high processing power and storage capacity, they are usually responsible for the storage of data that is captured and transmitted by devices. For this reason, it is used the concept of Feeds (system risers). These feeders are a specific part of the API that works with the reading and writing of data from the system.

Each Feed is a set of channels (datastream) that enables the exchange of data between the APIs and authorized devices. These channels are designed by programmers to separate the data by specific characteristics. In view of this, it is possible to create public and private channels. The first are those that can be viewed and changed by all according to the BCC License, already the second, it is those whose access is permitted only to developers and those whose admission is granted. Within channels there is the concept of DataPoint which is the representation of the data in a given time (timestamp) (Xively, 2013).

3 RELATED WORK

In this section will be elucidated the main existing Web-APIs and what features they have. These features are related to the requirements that were previously seen.

3.1 ThingSpeak

ThingSpeak is an API for "Internet of Things" open-source that stores and retrieves data from devices using the Hypertext Transfer Protocol (HTTP) over the Internet or simply of a LAN (Local Area Network). With this API (Application Programming Interface) it is possible to create applications in sensors for data records in a given environment, tracking, location and social networks of "things" (ThingSpeak, 2013). The data manipulation occurs by means of its channels, which have eight fields to be fed with data numeric and alphanumeric pagers, in addition to fields such as latitude and longitude, elevation and status.

3.2 NimBits

It is a collection of software components designed to record data of time series, such as for example, the changes in temperature read by a given sensor (NimBits, 2013). This API has the drive of events (triggers) during the recording of data. In This way, it is possible to perform calculations or trigger alerts along with your receipt.

Another advantage is that it was designed to be the first historian of world data, which means that you can download it and install it on any server, local or in the cloud, so that it is used the Linux Ubuntu and Google App Engine. This approach allows all instances of API to relate being possible to find other feeders (feeds) of data and make possible a connection with them (NimBits, 2013).

3.3 Open.sen

Currently in beta stage, this tool allows a rich visualization of results, so that by SenseBoard, you can see the incoming data in real time. The SenseBoard is powered by applications that are developed and installed within the API itself. These applications are independent but can be easily integrated with feeder (feeds) devices (Open.sen., 2013). These feeders communicate with the API through channels that are connected to devices. Even so, it is possible to capture information from other applications, not needing a direct contact with the device.

3.4 Cosm

This tool (formerly called Pachube) was developed to be a platform as a service (PaaS) for the Internet of things. With it, you can manage multiple devices through the RESTful resources, thus it is possible to deal with all the components of the API (Feeds, triggers, datastreams and datapoint) using commands via HTTP URLs, as already seen, PUT, GET, DELETE, POST. PUT is used to change the data, GET to reading, DELETE to erase and POST to create resources to communication or control (Xively, 2013).

3.5 SensorCloud

The SensorCloud is a tool storage for sensors "things". SensorCloud provides a Rest API to allow the upload of data to the server. The API implementation is based on patterns of HTTP commands. Soon, it is easily adapted to any platform (SensorCloud, 2013).

The communication between the tool and the sensors is totally on top of HTTPS, which means the entire communication between the channels and the devices are encrypted.

The format for sending and receiving data is the XDR (External Data Representation), it is not yet possible to use the templates known JSON and XML. The purpose of not using the formats standards of delivery is due to the fact that the XDR is not text but binary mode and with this it is possible that sensors for low processing power are able to send larger amounts of data than the standards based on text (SensorCloud, 2013).

Their components are divided hierarchically, where the device is at a higher level and contains the sensors that are divided into channels and these have the data.

3.6 Evrythng

It is a platform for powering applications or services directed by dynamic information about physical objects. Your goal is that all things must be connected, thus sets a world where all 'Thng' have a digital presence of assets on the Internet, even in social networks if desired, allowing the rapid development of Web applications using real-time information flowing from, any object in the world (Evrythng 2013).

3.7 iDigi

It is a platform in the cloud for managing network devices. It offers management gateways and endpoints on the network. It presents security policies of leaders in the industry, and great scalability for the exponential growth of devices on the network (Etherios, 2013).

3.8 GroverStream

GroveStreams is one of the most powerful platforms in clouds capable of providing real-time decision making for millions of users and devices. Among several of its qualities is the code generation per device. In this API it is possible that when you choose your device and the function that it will play a code that can be used to synchronize the device with the API is generated, thus there is only a need to copy this code paste in compiler used by the device and send to motto for which the code was generated (GroverStream, 2013).

3.9 Comparison of Web-APIs

Until the moment when it was explained the main features of Web-APIs, however as it can be seen in Table 1, there are some gaps that are still not handled by any of them. One of these characteristics is the autoconfiguration. Although, many Web-APIs provide examples of codes for configuring devices, none of them provides a side focused for the weak link in the IoT, the motes.

4 PROPOSED ARCHITECTURE

The mechanism proposes an architecture that addresses two types of problems. The first is the reduction of the complexity of devices' initial configuration, currently this setup is done through the provision of source code on the part of the Web-API and the compilation and deploying into the device by the user that is configuring the network. Then there is he must to go to the Web-API and configure it to receive the data according to the settings that were made available for the device.

The second problem is the reconfiguration of device that have already been configured and deployed on the network. Actually, if it is required reconfigure some devices that are already doing their job, the user needs to go up to the place where the mote is and remove it from the network to perform again the setup process, passing again by the first problem that was quoted.

The proposed mechanism will be divided into two parts: CLIENT and SERVER. Figure 1 provides a representation of the architecture.

The CLIENT's side Web-API will be native in motes and will be developed using techniques for low power consumption and memory. Initially intends to use the C language that can be compiled in almost all devices and techniques of concurrent programming and events: Protothread. The project will be carried out using Contiki, because in addition to being lightweight and perform the treatment of energy control of wireless transmitters through the ContikiMAC, it can also be run in almost all existing devices.

Thus, the CLIENT will provide to the SERVER's side the initial information of itself: what it is, what type of service it offers, what pin he uses to read, etc. For example, to connect a device into a network, it goes in search of Web-API for which was previously configured. When found, it sends the same to the information it holds about the device on which it is hosted. For realization of this

Table 1: Characteristics of Web-APIs.

	Open-Source	REST	Markup language data	Centralized Architecture	Security	Self Configuration	Provides Code for the device configuration	Cloud Storage
ThingSpeak	x	x	x	x	x		x	x
Nimbits	x	x	x	x	x		x	x
Open.sen		x	x	x	x		x	x
Cosm		x	x	x	x		x	x
SensorCloud		x		x	x		x	x
Evrythng		x	x	x	x		x	x
iDigi		x	x	x	x		x	x
Grovestreams		x	x	x	x		x	x

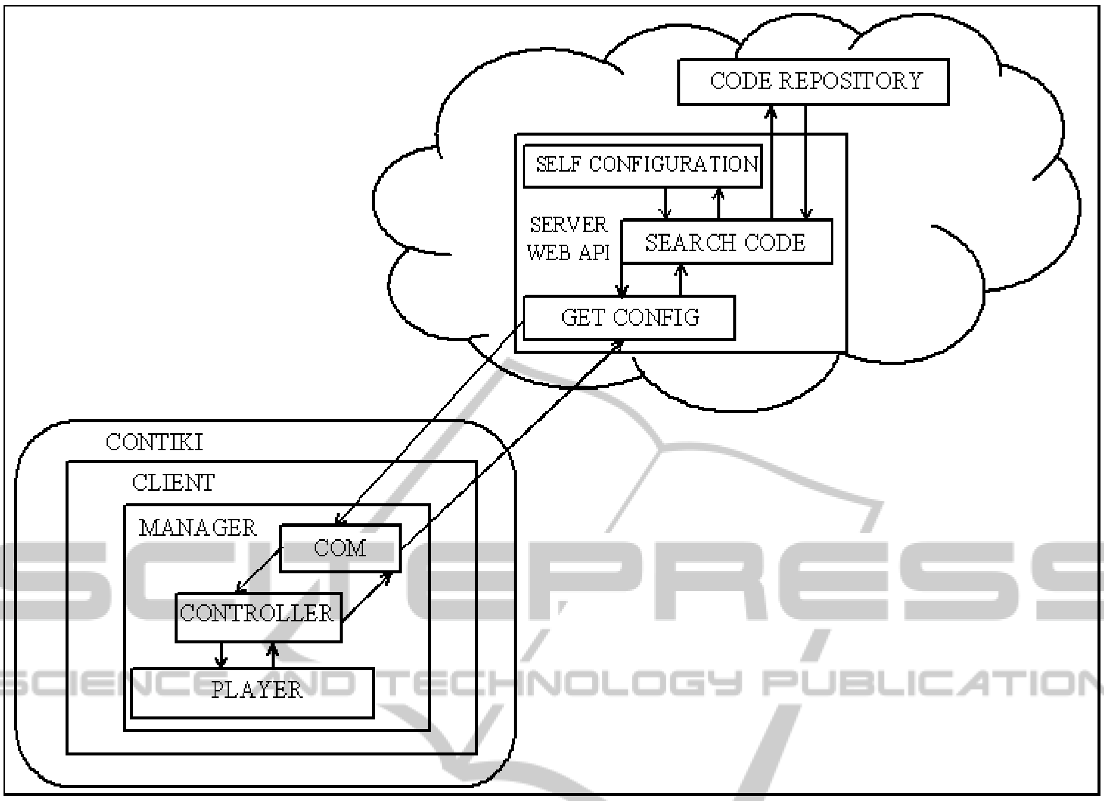

Figure 1: Proposed Architecture.

communication standard message format is also proposed.

The CLIENT is divided into 4 parts:

- Responsible for run of the code (Player);
- Responsible for version of configurations that are rotating (Controller);
- Responsible for sending and receiving of configuration data (COM);
- General monitoring of all parties (Manager).

After receive the source code COM passes for the CONTROLLER that will check the version and validations necessary for the implementation of the code, if it is accepted it goes to the PLAYER who will run the code received. If not, it creates an invalid code message and returns to the COM, which will send to the SERVER.

The Web-API that receive the information will check the authentication of this device (APIKEY) and the type of configuration that it needs to be able to carry out their functions.

As there are many repositories available for devices within the Internet, as the github, the initial idea is to go after the existing codes and transfer them to the requesting device. Thus in order to go search code of a specific device, the WEB-API first must receive a GET with the specific characteristics of the device. Upon receiving the search engine of the desired settings it returns to the mote, which will start the process of self-configuration.

When the configuration that will be passed to the device is discovered, the Web-API will also need to self-configure, providing new features to the code that is being executed by the motes can send the information. Even so, through these features developers can also configure new applications in the cloud that can promote the environment, such as sharing data between Web-APIs, and integration with social networks.

5 CONCLUSIONS

The architecture presented will initially provide two benefits: First is the development of techniques that may be useful for systems that want to implement the self-configuration.

The second one is the own development a tool using the architecture, once implemented this mechanism, it will assist in setting up a computers' network that works with wireless sensors and IoT. Thus, if the focus is the analysis of new systems of sensors, configuration will no longer be a complicated and a time consuming step, letting all the attention be directed to the main objective of the research.

For future work there is the possibility of developing the proposed architecture using existing Web-APIs, stated that work like ThingSpeak and Nimbits are great candidates for this development since they have available for IoT good features and they are open-source.

Another challenge is the development of a Web-Api that will provide the self-configuration for other existing Web-APIs. With this, besides the implementation of the proposed architecture for IoT, will need to draw up a technical architecture and interoperability for the Web-APIs of the Internet of Things and the clouds.

REFERENCES

Atzori, L; Iera, A; Morabito, G. The Internet of Things: A survey. Computer Networks, 2010. Computer Network.

Bandyopadhyay, S. A. Survey of Middleware for Internet of Things. *International Journal of Computer Science & Engineering*. Survey (IJCSES), 2011.

Zeng D., Guo S, and Cheng Z. The Web of Things: A Survey. *Journal of Communications, vol. 6*, setembro 2011.

https://xively.com/dev/docs/api/security/keys/ last accessed October 14, 2013.

http://www.json.org/ last accessed October 14, 2013.

http://www.w3.org/XML/ last accessed October 14, 2013.

http://www.computerworld.com/s/article/43487/Application_Programming_Interface last accessed October 14, 2013.

https://www.thingspeak.com/ last accessed October 14, 2013.

http://www.nimbits.com/ last accessed October 14, 2013.

http://open.sen.se/apps/29/ last accessed October 14, 2013.

http://www.sensorcloud.com/sites/default/files/SensorCloud_Open_Data_API.pdf last accessed October 14, 2013.

https://grovestreams.com/ last accessed October 14, 2013.

http://www.etherios.com// last accessed October 14, 2013.

http://www.contiki-os.org/ last accessed October 14, 2013.

Ruane, Laure. Protothread. UserGuide. https://code.google.com/p/protothread/wiki/UsersGuide. 2013 last accessed October 14, 2013.

Parachar M, Hariri S. Autonomic Computing: An Overview. *Springer Berlin Heidelberg*. 2005.

Kephart, Jeffrey O. The vision of Autonomic Computing. *IEEE Computer Society*. 2003.

http://www.evrythng.com/ last accessed October 14, 2013.