Available online at www.sciencedirect.com

ScienceDirect

Systems and Software 00 (2016) 1–4

Systems and Software

A Holistic User Experienced Management System for Internet of Things Networks

Claudio A. S. Wunder, Caio C. M. Silva, Rafael T. S. JR

Abstract

Keywords:

1. Introduction

2. Related Concepts

3. Proposal

In this section it is presented the IoT management system approach. Figure 1 presents the proposed architecture. The system architecture was conceived into five layers, denominated, user experience layer (UXL), presentation layer (PL), input interpreter layer (IIL), business logic layer (BLL), and communication layer (CL).

Broadly speaking, the UXL was idealized to make possible an IoT management through a natural interface. In the other hand, the PL receive a HTTP request, which represents a solicitation of a system feature, and routes to the respective feature handler. The IIL digest and solve the received data from PL and invoke the respective system action. The BLL receive the invoked action from IIL and execute the solution to complete the required feature. The CL does the communication using a REST approach as indicated in [1], sending requests to a service layer.

The proposal presentation is divided into four subsections, where each section describes each architecture layer of the proposed system.

3.1. Presentation Layer

The presentation layer provides a full aware IoT network management context.

3.1.1. Routing Module

The routing module is a binary tree router, that works by defining routes as tree nodes, applying validations by the patterns defined by the *Nodes* presented in the system. The RM works by using a binary search algorithm. The node is identified by a ”two-step” validation, which the first step is validating the input by a *REGEX* pattern, and the second step is an input validation inside the specific Node Handler. Which can result in true or false, depending of the node validation. The validation system also applies a sensitivity algorithm that finds false positives matches in the router, by applying a pattern recognition algorithm. The false positive nodes are excluded from the tree after a recall. After the execution of the algorithm, the matched node invokes the correspondent Input Handler from the section 3.2. An example of an addition of a node in the system is defined in the listing 1.

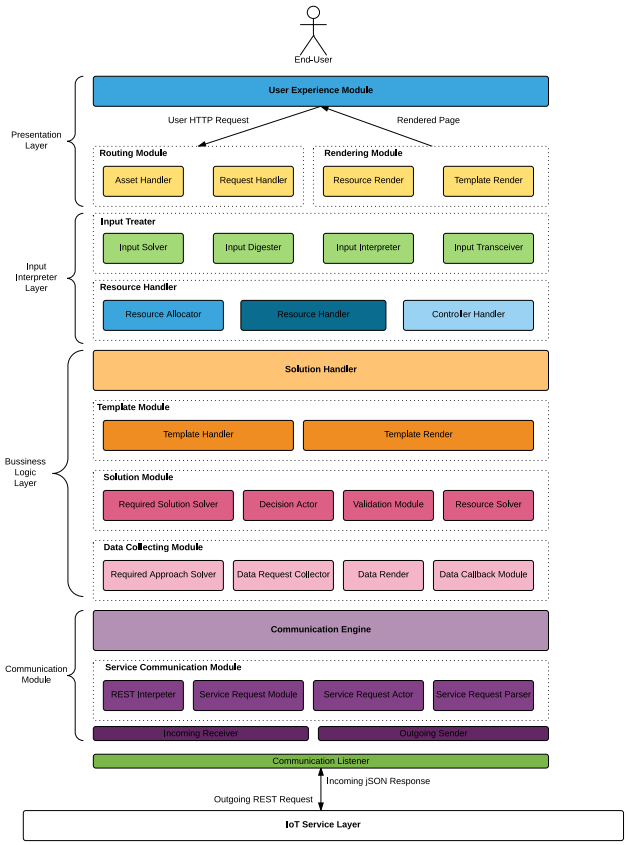

Figure 1. IoT management system architecture.

3.1.2. Rendering Module

The rendering module is responsible of allocate and invoke the Templates, Web Assets - like as Cascade Style Sheet, Java Script and Image files - for the desired solution. The set of those data is called in the system as ”Asset Resource Group”, for IoT requests these sets are defined by the interpolation of the requested HTTP method and of groups of data defined by the Business Logic Layer, described in the section 3.3.

3.2. Input Interpreter Layer

The IIL is liable to understand the requested resource. In order to interpret the submitted request, IIL performs the process presented in Figure 2.

The figure process starts when PL sends a resource request to IIL. The request data is received by Input Solver which solves and map it.

In the next step, a raw data map is sent to Input Digester, which cleans the data, removing useless and uncomprehensible information. If the map does not represent a valid set of information, the data is discarded and the process ends. In other way, the map is redirect to Input Interpreter which associate the data map elements into valid resource elements. Finally, the map is treated in Input Transceiver, joining and associating similar resources.

Thus, the resource map is routed to Resource Handler Module, which allocate the specific resources into memory. In addition, the resource handler performs all invocations related to it correspondent resource.

The following sub sections intents to explain each component of input interpreter layer, presenting their respective functionalities and limitations.

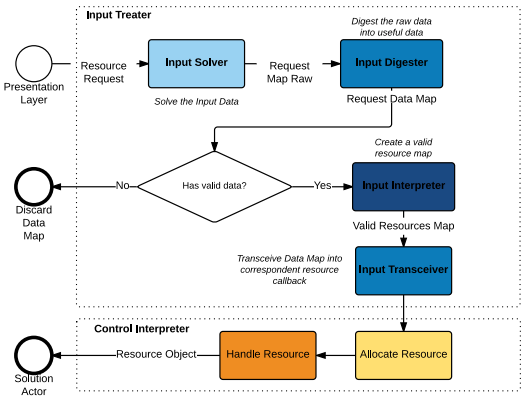

Figure 2. Input Interpreter Layer Flow Chart

3.2.1. Input Treater

The IT has four components, input solver, input digester, input interpreter and input transceiver. The input solver and digester use predefined regular expressions to identify which input expressions are valid. Those validated expressions are assigned to a correspondent route.

For an example, the Listing 1 shows a route being created. The function *addRoute* receives five parameters. The first one indicates the allowed regular expression for the route. The second one, set the type of node that will be invoked. Moreover, the third one defines the route segment level. Finally, also is defined the RESTful method, and the public route name.

Listing 1. Route Assign

```
Router::addRoute('path', new SomeNode,
    Deepness, Http::GET, 'category');
```

After the route assign, the input interpreter receives the resource map, which identifies the correspondent resource callback for each data, ignoring information that does not belong to any existing resource. Thereby, a full comprehensible resource map is obtained, containing valid resource information. The resource map elements are simplified by input transceiver, which joins resources of the same type. Finally, the resource map is sent to the resource handling layer, which is explained in Section 3.2.2.

3.2.2. Resource Handler

The resource handler module defines which type of Resource is being requested. A Resource, in the view of the system, is a specific category of data for the IoT network. Those categories are available through the IoT Service Layer. And only the SL knows the entire category and its reason of existence. The presented system only will know the characteristics of the Resource and its content. And only will know if is the ”wish” of the user/client. The RL is a generic software factory that creates in execution time a definition of a Resource and delivering it to the Business Logic Layer. Also the RH is responsible to allocate hardware resources and to ensure that the system can handle the requested Resource.

3.3. Business Logic Layer

The business logic layer works as a solution solver engine by applying the refined input from the IIL and execute a specific solution by a group of patterns and data. The BLL uses the required HTTP method, the refined input data from IIL and the requested Resource meta-data, applying a combination of steps explained in the next section.

3.3.1. Data Collecting Module

The Data collecting module works by collecting the necessary data to interpolate which solution need be taken. The desired Data Collector is defined by the steps of the IIL, with a DCM defined, the module start a conversation

with the IoT Service Layer that asks a group of specific sets of Data. Those questions that are made to the SL contains responses that fills the requested solution necessary data, those responses contains the meta-data of the requested resource, the characteristics of the resource, and depending of the requested solution also the data correlated to the resource. In the end the DCM creates a set of the data and transfer it to a Solution Handler.

3.3.2. Solution Handler

The solution handler works with two sub modules, those are the ”Data Treater” and the ”Data Handler”. The job of the SH is interpolate the collected data from DCM and take a specific solution. The solution is the combination of the work of a Data Treater and a Data Handler, where the last one does the equivalent of a MVC Controller, creating the final template that goes to the View. A Data Treater consists in solve the collected data, applying patterns to identify which solution need be taken. The available solutions are defined in the system, and represented by system classes. A Data Treater applies an TF-IDF algorithm to solve which Handler need be invoked, and after that a Data Handler parses the data.

3.3.3. Template Module

The template module consists in define the requested template by a Score Algorithm which identify the best template for the requested solution. The TM defines a Template and a View for the solution also doing the job of the union of the output of a Solution Handler and the combination of Template/View. The result is delivered for the user/client.

3.4. Communication Module

3.4.1. Communication Engine

3.4.2. Service Communication Module

3.5. Communication Interface

Table 1. *Caption* comes before the table.

	title page	odd page	even page
onesided	leftTEXT	leftTEXT	leftTEXT
twosided	leftTEXT	rightTEXT	leftTEXT

3.6. User Experience

4. Experimental Environment and Result Analysis

5. Conclusion and Future Work

Acknowledgments

The authors wish to thank the Brazilian research, development and innovation Agencies CAPES (Grant FORTE 23038.007604/2014-69), FINEP (Grant RENASIC/PROTO 01.12.0555.00) and the National Post-Doctorate Program (PNPD/CAPES) for their support to this work.

References

[1] C. C. de Melo Silva, H. G. C. Ferreira, R. T. de Sousa Júnior, F. Buiati, L. J. G. Villalba, Design and evaluation of a services interface for the internet of things, Wireless Personal Communications 1–38.