Poster

Automatic Allocation of Tasks in T-Res for WSN

Shashank Gaur
Luis Almeida
Eduardo Tovar

CISTER-TR-190604

Automatic Allocation of Tasks in T-Res for WSN

Shashank Gaur, Luis Almeida, Eduardo Tovar

CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail:

https://www.cister-labs.pt

Abstract

This paper presents a demo of an extension developed to support an existing programming abstraction for IoT: mT-Res. mT-Res is an extension of the T-Res programming abstraction, which allows users to write applications using a web framework without low level knowledge of resources. The paper describes an automated mechanism for allocate resources to such applications and adapt to changes in those resources.

Automatic Allocation of Tasks in T-Res for WSN

Shashank Gaur[1], Luis Almeida[2], Eduardo Tovar[1]
[1]CISTER/ISEP, Polytechnic Institute of Porto, Porto, Portugal
[2]FEUP, University of Porto

Abstract

This paper presents a demo of an extension developed to support an existing programming abstraction for IoT: mT-Res. mT-Res is an extension of the T-Res programming abstraction, which allows users to write applications using a web framework without low level knowledge of resources. The paper describes an automated mechanism for allocate resources to such applications and adapt to changes in those resources. This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UID/CEC/04234).

Author Keywords. wireless sensor network, cyber-physical systems, macroprogramming, applications, adaptation, internet of things, context-awareness

Introduction

Programming abstractions have been a major focus for wireless sensor networks (WSNs). With the evolution of hardware and software technologies, WSNs have become an integral part of the Internet of Things (IoT). Such evolution allows IoT to leverage heterogeneous hardware available in the network to a much larger extent than simple WSNs. However, this increases complexity in designing programming abstractions for IoT. T-Res is a recent important technological contribution in that direction.

Applications in T-Res are created through a structure called T-Res tasks. The T-Res task is the combination of four sub-resources: *Input Source (is)*, *Output Device (od)*, Processing Function (pf), and *Last Output (lo)*. The main application logic is stored in the */pf*. The URI addresses of Input and Output devices are stored in */is* and */od* respectively. The most recent output of the application is stored in */lo*. The complete T-Res task can be hosted on an IoT device and Constraint Application Protocol (CoAP) procedures are used to configure the sub-resources. With such a structure T-Res establishes separation between data processing, input, and output. In the next section, we examine an example to understand the need for our extension in T-Res.

mT-Res

Let us look at a simple application in T-Res. This application keeps the temperature of a room between a fixed range of 19°C and 22°C. Let us assume there are two temperature sensors and one heating actuator inside the room. To deploy this application using T-Res, a T-Res task must be created, as shown in Figure 1A. Initially, the T-Res task can be hosted on one of the temperature sensors. The */is* can be the temperature sensor other than the host. The */od* can be the heating actuator. The */pf* can be set to a script which takes the input from */is*, performs the calculation to check the temperature bound and provides the output instruction to */od*.

Now assume after some time the temperature sensor set as */is*, is no longer available due to energy failure, communication cost, or any other change in the IoT. Since there is another temperature sensor available in the room, the system should be able to detect this change, adapt to it and then use the other sensor as */is*. However, in T-Res, that must be done by the user by providing manual instructions using a CoAP agent. We have designed and implemented an extension, called mT-Res, which facilitates above-mentioned capabilities on top of T-Res.

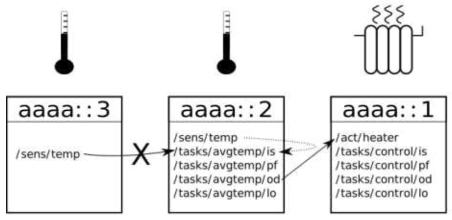

Figure 1A: T-Res Tasks

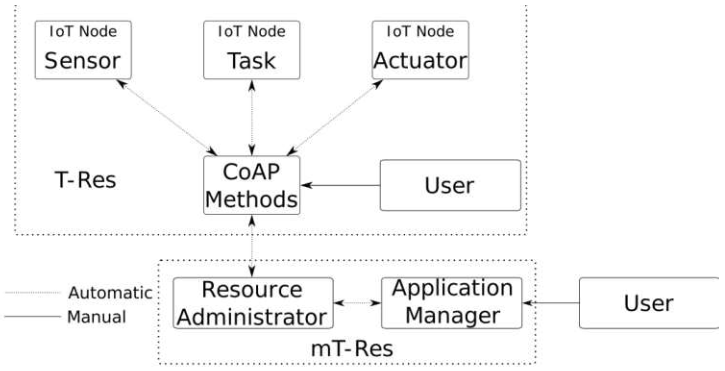

Figure 1B: mT-Res Design

mT-Res is divided into two parts, Resource Administrator and Application Manager, as outlined in Figure 1B. As of now, we keep both parts centralized in the system. The Resource Administrator deploys the code to host devices, assigns the input and output devices and keeps track of any changes in the system.

For user to create a new application, the Application Manager provides a Django enabled web-interface. This interface contains four fields, similar to the T-Res Task structure. These are input, output, host, and code. In the code field, the user can provide the same code as required for T-Res. In the Input/Output/Host fields the user can do an abstract selection by selecting the type of resource the user wants to use. The user does not need to provide URI addresses of specific resources.

Once a user submits the application, the Application Manager utilizes Resource Administrator to check available resources and deploy the application accordingly. The Application Manager creates a list of acceptable resources for each allocation. If any of the lists are empty, the framework notifies users that the desired resources are not available. Next, it will choose one of the options from each list at random, assigns a category "Active" to selected resources and "Available" to remaining resources. Once the selections are available, it requests Resource Administrator to do the allocations.

The Resource Administrator is enabled via python scripts, which provide automated CoAP operations (such as PULL, PUSH, GET, and OBSERVE), in form of python functions. Using these operations, Resource Administrator keeps track of all resources. If any operation returns with an error, the Application Manager will once again execute the process to allocate resources. However, this time, it will use another available resource for the corresponding error received earlier.

Conclusions

We extend capabilities of T-Res to provide autonomous resource allocations for IoT applications. In addition, mT-Res provides a web-interface for user(s) to input applications independent of specific resources. This extension is an effort to support context-aware IoT

References

Alessandrelli, Daniele, Matteo Petraccay, and Paolo Pagano. "T-res: Enabling reconfigurable in-network processing in iot-based wsns." *2013 IEEE International conference on distributed computing in sensor systems*. IEEE, 2013.

Gaur, Shashank, Raghuraman Rangarajan, and Eduardo Tovar. "Extending t-res with mobility for context-aware iot." *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2016.