Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 52 (2015) 29 – 34

Procedia Computer Science

The 6th International Conference on Ambient Systems, Networks and Technologies (ANT 2015)

Computations on the Edge in the Internet of Things

Andreas Moregård Haubenwaller[a], Konstantinos Vandikas[b]

[a] *Uppsala University, Uppsala, Sweden*
[b] *Management and Operations of Complex Systems, Ericsson Research, Stockholm, Sweden*

Abstract

In the Internet of Things (IoT), many applications focus on gathering data which can then be processed and visualized. However, such computations are usually spread generically based on parameters such as CPU and/or network load. This may mean that a significant amount of data needs to be transported over the network (either directly, or transparently using a network file system) in order for the data to be available to the node that is responsible for processing them. This paper proposes a method for deploying computations that can take factors such as data proximity into consideration. Thus, processing can be moved from central high-powered processing nodes to smaller devices on the edge of the network. By doing this, costs for gathering, processing and actuation can be minimized. In order to capture data dependencies among computations, but also to deploy and handle individual processing tasks in an easy way, the actor-model programming paradigm is used. To minimize the overall cost and to handle extra factors that weigh in on the distribution of tasks, a constraint programming approach is used. The combination of these two techniques results in an efficient distribution of tasks to processing resources in IoT. Taking into consideration the NP-hard nature of this problem, we present empirical results that illustrate how this technique performs in relation to the amount of devices/actors.

Peer-review under responsibility of the Conference Program Chairs
Keywords: iot; constraint-programming; actor model; edge computing

1. Introduction

By the year 2020, major technology companies expect that the number of connected devices will number in the range of 25-50 billion. Cisco and Ericsson believe that the number 50 billion devices will have been reached by 2020 [1,2]. The Gartner Group on the other hand expects that number to be around 26 billion [3]. Despite the large disparity in numbers, they all see a huge increase in the number of connected devices. As more and more devices are being connected, new uses and markets are springing up. Some examples would be smart homes and home automation, environmental monitoring, and smart cities. All of these devices connected together coin the term ”Internet of Things”. Internet of things, or IoT, is currently getting a lot attention, for example it is on the peak of Gartner’s Hype Cycle

* Contact author. Tel.: +46 725398073
E-mail address: konstantinos.vandikas@ericsson.com

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2015.05.011

for 2014[4]. Along with this hype and attention, several platforms specifically aimed for gathering and visualizing IoT data were introduced, for example, SicsthSense[5], Xively[6], SensorCloud[7] and the IoT-Framework[8].

The main focus of these platforms is to gather data from a wide variety of devices found in IoT so that it can be analyzed, visualized and possibly acted upon. These devices can be anything from small wireless nodes to large servers. However, IoT devices as such are commonly seen as smaller and less capable than generic home computers or servers, some common examples would be; Raspberry Pi[9], Arduinos[10] and Intel Edison[11] boards. The normal approach is to gather data from all of the different devices by either a push(the devices initiate the sending of data) or a pull(the platform will ask the devices for data) approach. After data has been gathered it can be stored and analyzed by the platform.

This paper proposes an approach where the data will be processed by computationally capable IoT devices rather than being sent to a central location for processing. The processing tasks to be carried out, will be split up and deployed on these devices based on a certain cost. The goal is then to minimize this cost in order to find an efficient deployment of tasks on to devices. Depending on what cost metric is used, different costs can be minimized, for example one might want to reduce the overall amount of data, flowing through the network or one might strive to reduce response times. This will be accomplished by using a constraint programming model to minimize the overall cost while also being able to be extended with other constraints that may impact the search space of possible deployment alternatives.

2. Design

The idea behind this paper is to find a way to efficiently use the processing resources available in many devices connected to the Internet. Platforms in which you can register devices or data streams and then gather and visualize the data, already exist. The question is how to evolve such platforms in order to gain access to the processing capabilities of previously under-utilised computationally capable devices. Some of the aforementioned platforms might already support features such as triggers on data and actuation, however, processing is done centrally. By moving the processing to devices that are closer to the source of the data, the response times for actuation can be reduced. If the platform is using a pull type approach when it comes to gathering data, then some of that functionality could be moved to nearby devices so that data can be packed together and compressed before being sent which would reduce overall network traffic. If the platform is using a push type approach, then the destination of the data can be changed from the platform to a nearby device so that data can be packed and compressed.

2.1. Tasks and Task Graphs

Data flow can be seen as a task graph, where data originates from different devices; data can then be aggregated and possibly reduced in size via generalisation. Subsequently, data is sent along for further processing, storage and also possibly acted upon. When the data is acted upon, it would most likely cause actuation in a nearby device. If the central repository is far away, then the data has to travel a very long way; First it travels from the device with the sensor all the way to the repository where it is processed, and then it is returned the same way for the actuation. In order to get rid of this behavior, the data flow or task graph needs to be split up. Splitting the task graph up or rearranging is quite easy. However, splitting up complex software programs can be a daunting task. Chances are that these tasks are not represented as individual tasks at all in a large code base such as an IoT platform which would require quite a lot of code to be rewritten. Thus, if the program was already split up into smaller pieces then it would be a lot easier to ”break out” individual tasks and place them on different locations(devices). With this in mind, the actor based programming model[12] seems like a very good fit for this proposed IoT solution.

2.2. Cost-Based Distribution

In order to distribute the actors to devices in the Internet of Things one needs to figure out on which device each actor should be deployed. This will be solved by assigning a certain cost to all connections between actors as well as a cost between all devices. When an actor is deployed on a certain device, then the actor cost will be multiplied by the device cost. By deploying all actors on to devices, we can get a total cost of the actor graph. This cost can then be minimized by choosing different devices on which the actors are deployed on. The deployment should also be able to