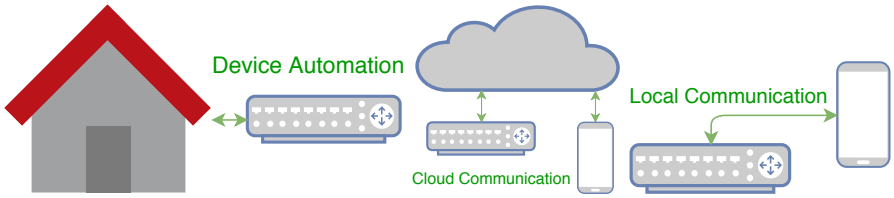
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2019
http://excel.fit.vutbr.cz

Autonomous Automation in the Internet of Things

David Piškula*



Abstract
The aim of this work is to assess the current state of automated Internet of Things networks, describe the problems of existing solutions and design a system that solves some of them. Two of the most common issues are the overdependence on Cloud servers and loss of functionality without an internet connection. The designed system solves these by moving automation from the Cloud to a gateway at the edge of the network. The gateway connects to the Cloud to report device states, store telemetry and receive remote commands, however, it is able to perform automation and data processing even in an offline state. The result is part of a complete Internet of Things solution used in a smart home model created in cooperation with NXP Semiconductors.

Keywords: Internet of Things — Edge Computing — Autonomous Automation

*xpisku02@fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Our world is becoming more connected every day. This resulted in the emergence of the Internet of Things, which brings internet connectivity to all kinds of objects from industrial machines to wearable devices, sensors and actuators. This area of technology is still relatively new however, and there are many problems that need to be solved before it can truly become mainstream. Among those problems is the insufficient autonomy of connected end points and their overdependence on Cloud servers explored in this paper. The purpose of this work is to design a system that:

- Leverages advantages offered by the Cloud
- Remains functional when disconnected
- Offers customizable automation schemes
- Prevents the loss of valuable data

The previous trend was to integrate the Internet of Things with Cloud Computing and leave most of the data processing to the Cloud. The papers [1] and [2] describe how the limited computational capacity of low-power Internet of Things devices is not enough to achieve on-site data processing. Additionally, they describe how these devices are not capable of collecting, storing and analyzing the large amounts of data generated by Internet of Things networks. Instead, they suggest a paradigm they call CloudIoT, in which all of the data is sent directly to some Cloud platform, processed there and any resulting configuration changes are then sent back to the devices.

The suggested approach, however, is vulnerable to Cloud service outages and brings latency to data processing and device automation. The aim of this work is to study and describe other approaches that have been surfacing in the last few years and based on them to create a working application as a demonstration of a possible solution to these issues.

Among home automation solutions, there have been several failing products on the market. Revolv [3] was a smart home hub with the purpose of controlling a wide range of different gadgets via a smartphone app. However, the hub was completely reliant on the developer company's cloud-based service. Revolv was

eventually bought by Nest and subsequently, the cloud service was shut down, rendering all Revolv hubs inoperational. Another product with a similar fate was Emberlight [4], a smart light socket designed to work with ordinary light bulbs. Its aim was to enable a mobile application to control regular light bulbs instead of having to buy specialized and expensive ones. The issue was that all controls were done though a Cloud service which was, again, shut down.

In the last few years, a new design paradigm has been gaining popularity - the Edge Computing. This strategy brings some processing power back from the Cloud to the local network. Such idea was explored in [5] with the goal of implementing a smart e-Health gateway to bring the Internet of Things to healthcare. The paper focuses on inventing a system architecture that would allow for a secure network with local storage, data filtering and analytics. The result of their work was UT-GATE, a functional gateway with a WebSocket server providing local servicing while communicating with a remote Cloud platform to receive improved processing rules gained through deeper analytics.

Another study concerning this area was described in [6]. This team of researchers tried to tackle the issues by proposing a hierarchical fog computing architecture that is flexible, scalable and brings computing resources close to end devices. The architecture consists of a network of computationally powerful fog nodes that can each connect to a Cloud server to offload their work if needed. The resulting design can substantially reduce traffic loads in networks and the communication delay that can be a problem in purely Cloud Computing based systems.

There are also products on the market already that offer partial or even full functionality to connected networks even without an internet connection. The latest Amazon Echo Plus, for example, allows limited usage of their voice controlled home control hub offline. Phillips Hue Bridge is another example, as it only needs an internet connection for remote control but locally can work completely offline. Hubitat [7] is a home automation platform, that was built with offline functionality in mind. Despite the ability to use an internet connection for updates and Cloud communication, it does not need the internet connection for any of its major functions. This makes the whole system more secure, private and removes latency.

The solution presented by this work solves Cloud dependency by performing data processing and automated actions on the network's gateway. It uses conditions that act as rule sets to decide which device should be affected and how, and reports state changes to the Cloud. These conditions can be set up either remotely through the Cloud or even directly, if a user with a connected mobile application is in range of the gateway's wifi or bluetooth signal. There are also processes in place, that take care of reconnecting the gateway automatically and resynchronizing states with the Cloud. Finally, telemetry reports can be buffered in several different buffering modes, which ensures that no important data is lost even with prolonged offline states.

The design was created with extensibility in mind. The conditions used for automation are based on JSON string processing which makes them very flexible and easy to understand. The gateway uses the widely popular MQTT protocol to communicate with the Cloud, therefore it is not bound to a specific Cloud platform. Thanks to the deterministic approach to reconnection and resynchronization, states and configurations are always in order. The ability to communicate and perform data analytics offline lowers latency and keeps the gateway and the whole network functional even if it disconnects from the internet.

2. Theoretical Background

This chapter talks about the Internet of Things and some of the technologies that enable it. It describes Cloud Computing, Edge Computing and the MQTT communication protocol to provide the reader with deeper understanding of some of the terms or phrases used in this paper.

2.1 Internet of Things

The Internet of Things is a phrase that has been very popular in the last decade but has been used to describe many different ideas. A recent paper about the understanding of the Internet of Things [8] goes through the evolution of this paradigm, from RFID-based Wireless Sensor Networks for telemetry gathering, data acquisition and supervisory control, to moving away from tag-centric solutions and towards providing simple objects with the capabilities to directly connect to the internet and lastly to a service focused ideology with Cloud Computing and huge amounts of collected data. They also briefly talk about the emerging Edge Computing a try to predict the future of the Internet of Things. The definition the authors gave based on their research is "a conceptual framework that leverages on the availability of heterogeneous devices and interconnection solutions, as well as augmented physical objects providing shared information base on the global scale, to support the design of applications involving at

the same virtual level both people and representations of objects."

2.2 Cloud Computing

As described in [9], Cloud Computing is an on-demand computing model composed of autonomous, networked IT resources. It enables organizations to leverage internet-based, scalable, inexpensive and easy-to-use pools of resources instead of having to maintain their own datacenters. Cloud platforms often offer a subscription based model that allows customers to only pay for the resources they actively use. The Cloud's architecture can be split into several domains, the Infrastructure as a Service that provides direct access to hardware resources, Platform as a Service which supplies development tools and administration platforms for the hardware and Software as a Service that delivers software applications that can be subsribed to.

2.3 Edge Computing

In [10], Edge Computing's advantages are listed as the option to process the massive data generated by devices at the network edge instead of transmitting them to the centralized Cloud. It can provide services with faster responses and greater quality and can be considered the future the Internet of Things infrastructure. They also describe the various approaches to Edge Computing that are being implemented today. Cloudlets, for example, are small-scale data centers located at the edge of the internet. They consist of resource rich computers providing powerful computing resources to nearby mobile devices with lower latency. They can serve as a middle point between the mobile devices and the Cloud. Mobile edge networking, on the other hand, is defined as a technology that provides Cloud Computing capabilities at the edge of a mobile network. This can be represented by applications running as a virtual machine on a powerful mobile edge platform. Finally, Fog Computing aims to distribute resources and services along the continuum from the Cloud to things, using more powerful end nodes where possible.

2.4 MQTT

MQTT is defined in [11] as a client-server publish/-subscribe messaging transport protocol that is light weight, open and ideal for Machine to Machine communications and the Internet of Things because of its small footprint. It runs over TCP/IP, communicates through a broker application that provides one-to-many message distribution and offers three qualities of service. These are the QoS 0: At most once, that delivers messages to the best efforts of the operating environment, QoS 1: At least once, where messages are assured to arrive but duplicates can occur and QoS 2: Exactly once. MQTT can be secured through TLS with SSL Certificate exchanges between the broker and subscribers/publishers or with WebSockets.

3. Designing an Autonomous System

Several main goals were set for the design. It should provide automation through an easy to understand and extensible protocol, it should be as independent of the Cloud as possible while at the same time making use of the Cloud's advantages, like remote communication and storage, and it should be able to automatically reconnect to the Cloud and properly enter a synchronized state after any kind of connection loss.

To achieve these goals, the design was inspired by the Edge Computing paradigm. The most important part of that is computing all automation at the edge of the network. The automation scheme that was devised manages to do this by storing conditions at the gateway and based on them analysing all received data before passing it to the Cloud. Because of this, even if the connection to the Cloud is severed, the gateway can successfully keep controlling the network as needed. The solution is described further in subsection 3.1.

Another aspect that needed to be looked into was a mechanism that would prevent important data from being lost. The created mechanism is based on message buffering and was designed with customizability in mind, in order to give users control over their data. It was necessary to provide not only short term solutions for networks with frequent disconnections but also to take care of prolonged disconnected states in cases where data history is of great significance and the local storage is insufficiently large. The gateway handles these cases by reducing the amount of stored data through several different schemes that can be chosen separately for each device. The buffer schemes are discussed in subsection 3.3.

Furthermore, it was essential to maintain consistency in device states between the Cloud and the gateway at all times while connected and to develop a deterministic way of synchronizing these states after the gateway reconnects following a downtime. This is achieved through several mechanisms described in subsection 3.2 and subsection 3.3.

Lastly, to eliminate the inconvenience of not being able to manually control devices while disconnected from the Cloud, offline communication with a nearby mobile phone was developed as well. For this purpose, the gateway makes use of its WiFi and Bluetooth ca-

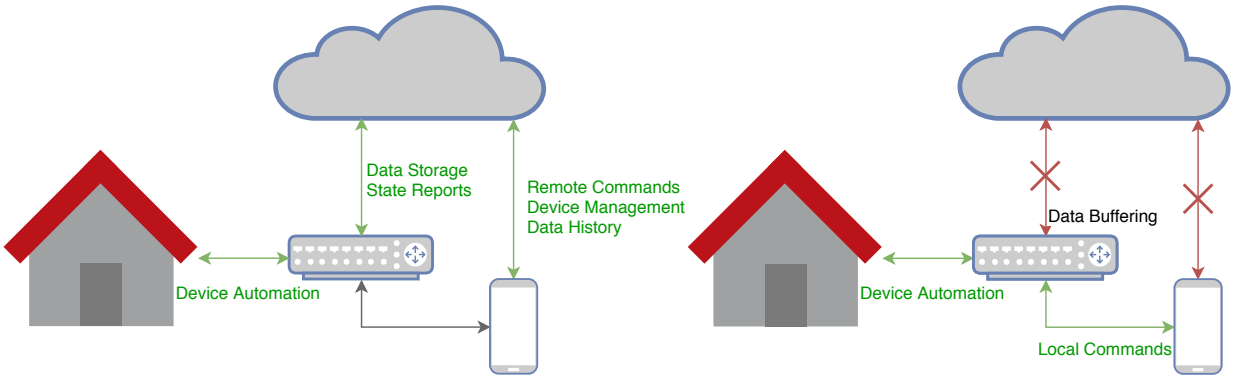


Figure 1. Functionality while connected and disconnected

pabilities to communicate with and receive commands from similarly enabled mobile phones. The result of this design choice is that the only cases where internet connection is absolutely necessary are the initialization of the gateway itself and the adding and removing of devices to and from the network.

To incorporate remote control, remote storage and pave the way for possible extensions to the design, the gateway application was envisioned with Cloud connectivity from the get go. This was approached by analyzing available Cloud platforms and making sure the design is as portable between them as possible. That's why the standard MQTT protocol was chosen for gateway-to-Cloud communication and why the gateway does not use any Cloud platform-specific APIs.

Figure 1 shows the differences in functionality between the connected and disconnected states.

3.1 Conditions

For the purposes of automation, a simple and extensible protocol was created utilizing the commonly used JSON strings. JSON provides a structured format that is readable by humans and is supported by many programming languages used today. Every condition has an owner, which is the device whose actions the condition is dependent on, a list of affected devices, the condition itself and the command that will be executed in case the condition is met. These JSON strings are then stored as part of the owner's state and every time a device reports data, all the device's conditions are examined for any commands that need to be executed.

3.2 Reconnection

Since MQTT was chosen as the communication protocol between the gateway and the Cloud, it is necessary to always maintain stable connections for every MQTT client representing a device inside the network. Whenever a client disconnects for any reason, a callback is triggered, that takes care of checking if the corresponding device is still alive and afterwards tries to refresh the client's connection. If the reconnection fails but the gateway is still connected to the internet, the client is marked by a background process that will periodically attempt to reconnect it at a later time. Finally, internet connection losses are taken care of by a process that listens to any connection changes that might occur and, in the event of connection recovery, is able to go through every MQTT client and reconnect them.

3.3 State synchronization and buffers

In order to avoid losing any important data, special buffers were created specifically for telemetry devices. These buffers feature several modes and are filled by messages whenever they cannot be transmitted. As soon as a client with a buffer reconnects, its buffer is emptied. When that is complete, the gateway processes the last command that was sent to the client through the Cloud platform, performs any required changes, reports the current device state and finally resumes normal functionality for the client.

The buffers operate in three main modes with one secondary mode. The main modes are HOLD, FIFO and DYNAMIC, the secondary mode is VARIANT. HOLD simply stores every message passed to it until it is filled and rejects everything after that. FIFO, as the name suggests, acts as a typical FIFO pipe and once filled, starts discarding the oldest messages in favour of new ones. DYNAMIC uses a time interval to space out accepted messages after it is filled up for the first time. When it becomes full again, it enlarges the interval by a specified amount of time. The VARIANT secondary mode can be applied to all the main modes and creates spaces between stored messages by making sure every pair's stored telemetry differs by a specified minimal value.

4. Experiments

In order to verify the functionality of the system, several experiments were performed in a controlled en-

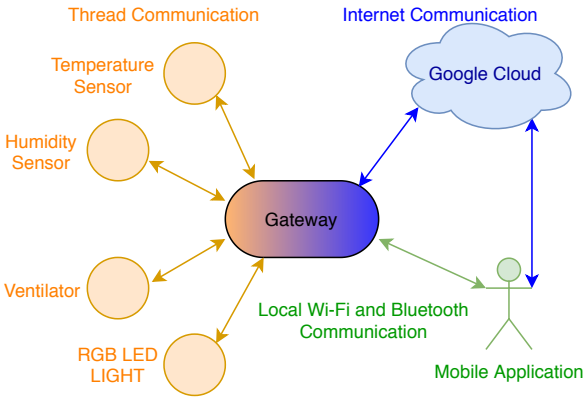


Figure 2. Topology of the experimental network

vironment. These experiments included the gateway, which was based on the Pico i.MX7 development kit with the Android Things operating system with a USB-KW41Z dongle for Thread communication, Google Cloud, an Android mobile application and four end devices using the KW41Z MCU inside custom-made modules created specifically for this project. The devices include a temperature sensor, a humidity sensor, a ventilator and an RGB led light. The software for the end devices was based on NXP's Thread SDK and the Android applications for the gateway and mobile device, as well as several processing javascript scripts running on the Google Cloud platform, were implemented from scratch.

A condition for each of the sensors was set up. The temperature sensor's condition required the gateway to turn the ventilator on or off depending on a temperature limit and the humidity sensor's condition instructed the gateway to change the color of the RGB led light depending on reported humidity. The conditions were simple for the purposes of the experimentation, however, in real world applications one could have a single sensor control a wide rage of other end devices simultaneously. One such example would be a thermostat that manages different ventilators or AC units and all of the heating inside a house.

First, the network was set up and left connected to test full functionality. The outcome of this test was that the mobile application could be used to control all end devices both remotely and locally while the gateway was also able to automatically control them according to the configured conditions. All reported temperature and humidity values were also stored in the Cloud and readable in graphs inside the mobile application.

Next the internet connection of the gateway was shut down. All subsequently reported data was still being analyzed and acted upon according to the already configured conditions but instead of states and telemetry being reported to the Cloud, the telemetry was buffered and state changes only marked in the gateway's storage. When the disconnection was detected in the Cloud, all devices were marked as offline in the mobile application. When using the local communication, the devices could still be controlled and new conditions could be added, but the data history was not being updated. When using remote control, the network was not affected at all.

Finally, connection was restored to the gateway. The buffers were emptied before any new data was reported, the last requested remote configurations were received and processed and at the end of the reconnection flow, the current states were reported. The devices were marked as online again and the system resumed full functionality, with the buffered data reflected in the data history graphs.

5. Conclusions

This paper researched and presented information about network autonomy and automation in the Internet of Things. It examined past and contemporary trends in this area, its problems and potential solutions and finally introduced the design and implementation of a new solution used in a smart home model.

The solution designed as part of this paper provides a simple yet effective condition schema that enables automation of networks. It performs all automation at the edge of the network and can be controlled locally without the use of internet. Furthermore, it connects to a Cloud platform to allow for remote control and telemetry storage.

The contribution of this work is a proof of concept model of a smart home that can be used to present the advantages of the Internet of Things and the technologies of NXP Semiconductors. It provides a viable solution to the overdependence of Internet of Things on the Cloud and offers an extensible autonomous automation scheme.

The resulting work can be further improved by adding machine learning capabilities by making use of the Cloud. Additional conditions can also be added to support a wider variety of devices. Since the design does not use vendor specific libraries, it could also be extended with connectivity to different Cloud platforms.

Acknowledgements

I would like to thank my supervisor Ing. Petr Musil and my colleague Ing. Karel Povalač, Ph.D. for their assistance and technical advice.

References

[1] A. Botta, W. de Donato, V. Persico, and A. Pescapé. On the integration of cloud computing and internet of things. In *2014 International Conference on Future Internet of Things and Cloud*, pages 23–30, Aug 2014.

[2] S. M. Babu, A. J. Lakshmi, and B. T. Rao. A study on cloud based internet of things: Cloudiot. In *2015 Global Conference on Communication Technologies (GCCT)*, pages 60–65, April 2015.

[3] Klint Finley. Nest's hub shutdown proves you're crazy to buy into the internet of things. *Wired*, May 4 2016. [Online; visited 2019.03.10]. Retrieved from: https://www.wired.com/2016/04/nests-hub-shutdown-proves-youre-crazy-buy-internet-things/.

[4] Fredric Paul. What happens when an iot implementation goes bad? *Network World*, November 21 2017. [Online; visited 2019.03.10]. Retrieved from: https://www.networkworld.com/article/3238004/internet-of-things/what-happens-when-an-iot-implementation-goes-bad.html.

[5] Amir-Mohammad Rahmani, Nanda Kumar Thanigaivelan, Tuan Nguyen Gia, Jose Granados, Behailu Negash, Pasi Liljeberg, and Hannu Tenhunen. Smart e-health gateway: Bringing intelligence to internet-of-things based ubiquitous healthcare systems. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 826–834. IEEE, 2015.

[6] Xiang Sun and Nirwan Ansari. Edgeiot: Mobile edge computing for the internet of things. *IEEE Communications Magazine*, 54(12):22–29, 2016.

[7] Hubitat elevation is a powerful home automation platform. [Online; visited 2019.03.10]. Retrieved from: https://hubitat.com/pages/home-automation-features.

[8] Luigi Atzori, Antonio Iera, and Giacomo Morabito. Understanding the internet of things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks*, 56:122–140, 2017.

[9] Qusay F Hassan. Demystifying cloud computing. *CrossTalk: The Journal of Defense Software Engineering*, 24, 01 2011.

[10] Yuan Ai, Mugen Peng, and Kecheng Zhang. Edge computing technologies for internet of things: a primer. *Digital Communications and Networks*, 4(2):77 – 86, 2018.

[11] Mqtt version 3.1.1. *OASIS Standard*, October 29 2014. [Online; visited 2019.03.11]. Retrieved from: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf.