IOSR Journal of Engineering (IOSRJEN) www.iosrjen.org
ISSN (e): 2250-3021, ISSN (p): 2278-8719 Special Issue || September 2021 || PP 22-25
7th National Conference On Advancements in Communication, Computing and Electronics Technology-ACCET 2021

Internet of Things: Over-the-Air (OTA) Firmware Update of Smart devices by Publish-Subscribe framework

Piyush Nindane
Electronics and Telecommunication Engineering
Modern Education Society's college of Engineering Pune, India

Bhagwat Suryawanshi
Electronics and Telecommunication Engineering
Modern Education Society's college of Engineering Pune,India

Shubhada Deshmukh
Electronics and Telecommunication Engineering
Modern Education Society's college of Engineering Pune,India

Abstract—*Internet of Things (IoT) has emerged for over a past few years and has been a substantial deployment on multiple architectures, standards, and platforms. Despite the heterogeneity of the devices, it must provide open access for the development of the existing firmware. This firmware must be kept in constant development to remove bugs and improve functionality. An over-the-air firmware update system is preferable as it allows for faster remote update and encourage accessibility for the development. This paper introduces a novel over-the-air (OTA) firmware update system based on PUBLISH-SUBSCRIBE model between a client-server communication, providing a low latency route discovery and establishment. Based on the investigation results, we demonstrate that OTA firmware update system gives fast update and easy accessibility and thus effective and beneficial for further use, i.e. smart urban development.*

I. Introduction

Internet of Things (IoT) has appeared as a communication paradigm of an integrated and interconnected various objects(things) with microcontrollers, transceivers, and suitable protocol stacks, which enable them to communicate with one another and interact with users to reach common goals. Different firmware are installed on these microcontroller boards to control the corresponding objects functionality and define its network communication. As increasingly advanced functionalities are developed, the need to update the running firmware of the embedded devices emerge. Updates can be used to improve existing functionality or to fix discovered bugs. One way to update a firmware is by performing over the air firmware update (FOTA). This method is preferable as it allows for faster update, encourage accessibility, and easy to implement and maintain. In this paper, we propose an over-the-air (OTA) firmware update system based on Publish-Subscribe model over peer-to-peer mesh (P2PMesh) architecture. The intended use of this architecture is to implement peer-to-peer application including route discovery and establishment, and data transfer at mesh topology that gives advantages of wireless link broadcast ability and reduction on redundant traffic and interference. On the other hand, this mechanism is a low-power wireless mesh network that support wide range of wireless connectivity.

Our system is composed of Sensor Node, Raspberry Pi as gateway, Package generator, and Package management (Repository) server all in LINUX environment and compatible with the Python software stack. These devices are arranged to form the P2PMesh with an addition of gateway node. We have used the TCP/IP as the main protocol and secured MQTT (Mqtt+TLS/SSL) service for the Pub-Sub modeling However for better security purpose we have used the FTP over SSH (SFTP) as a file transfer method. It is meant to segregate the abstraction of the internal evaluation we found that our system works well and suited to be implemented in both smart city, smart home, and many WSN development. The major contribution of our work is the development of a remote firmware update architecture with low-power energy consumption and efficient file dissemination.

The remainder of the paper is organized as follows. In
Section II we briefly discuss several work related to this system. Section III describes an overview of our system design. Section IV presents the implementation of our over-the-air firmware update system prototype and we conclude our discussion in Section V.

II. Related Work

As far as it concerns, we did not find much related work regarding an OTA firmware update prior to our research. D.K. Nilsson et al. [3][6] present a wireless firmware update over a hierarchical wireless network. The network is considered hierarchical since a single central unit is communicating with a number of end nodes. This paper utilized almost similar basic architecture, however, the need to develop a low-powered system is significant thus our system offer a mesh-based network protocol to facilitate communication between end nodes.

Al Asaad, et al. [4] provides P2PMesh architecture that implements peer-to-peer system on top of mesh topology for file sharing. The architecture consists of two subsystems: a mesh P2P subsystem runs on both gateway and end node, and a global P2P subsystem that runs at mesh gateway. The P2P subsystem is used to create an overlaying peer-to-peer on top of mesh network topology to support file exchange between gateway and end nodes, while global P2P subsystem used to set up TCP/IP peer-to-peer network connection with the internet.

III. System Design and Architechture

For system design, we utilize four different types of nodes involved in the system, namely End Node (Sensor Node), Gateway Node (as a Coordinator), Repository server (MQTT Broker), and Package generator, as will be explained as follows.

- *End (Sensor) Node*

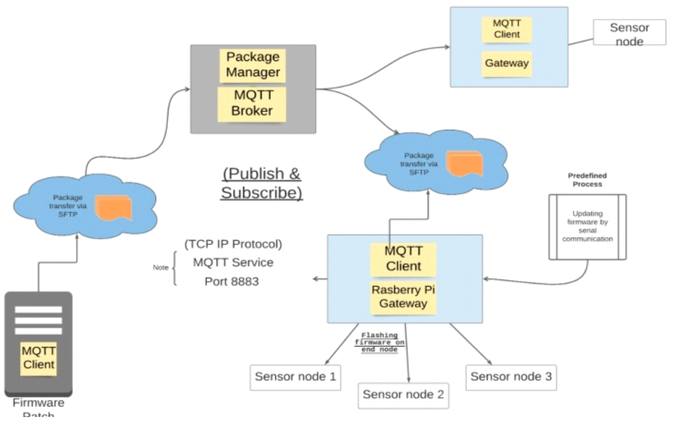
Define End node are devices that will execute the main functionality of the system. Each end node has the main capability to receive the firmware update data in its boot loader mode and execute the received firmware data in its application mode. As the system adopts P2PMesh topology, end nodes also provide the routing functionality that support the route discovery and establishment inside the network. The routing functionality is important, especially to facilitate the communication lane with the furthest end node inside the network that is unreachable by gateway. The whole system are depicted in Figure 1 which is divided into two major parts: external and internal network. External network is the usual TCP/IP-based connection in which a new firmware update version is prepared before being disseminated unavoidable.

- *Gateway Node*
- Gateway node which is a MQTT Client, is a connecting link between end nodes (Private network) and the external network connected with the well-known TCP/IP. It receives firmware update from the package manager server (MQTT Broker) and then transfer the data into the internal network.
- *Pakage manager*

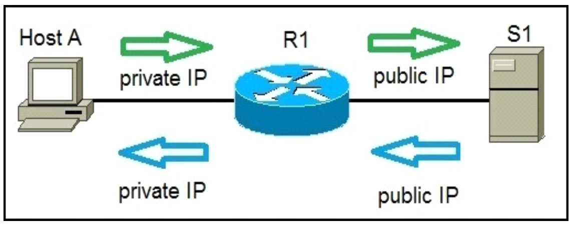
The package manager server is the central hub for communication between the original equipment manufacturer and the end users. As per the Pub-Sub model, this server has the MQTT broker running on it with ports open for SFTP service (Port 22). As the package is received by the manager the broker publishes an availability message to all the clients subscribed to a particular topic and in turn recieves an acknowledgement message from the clients followed by a SFTP pull (GET) request.

IV. Process flow and Conclusion

- *Network Architechture*

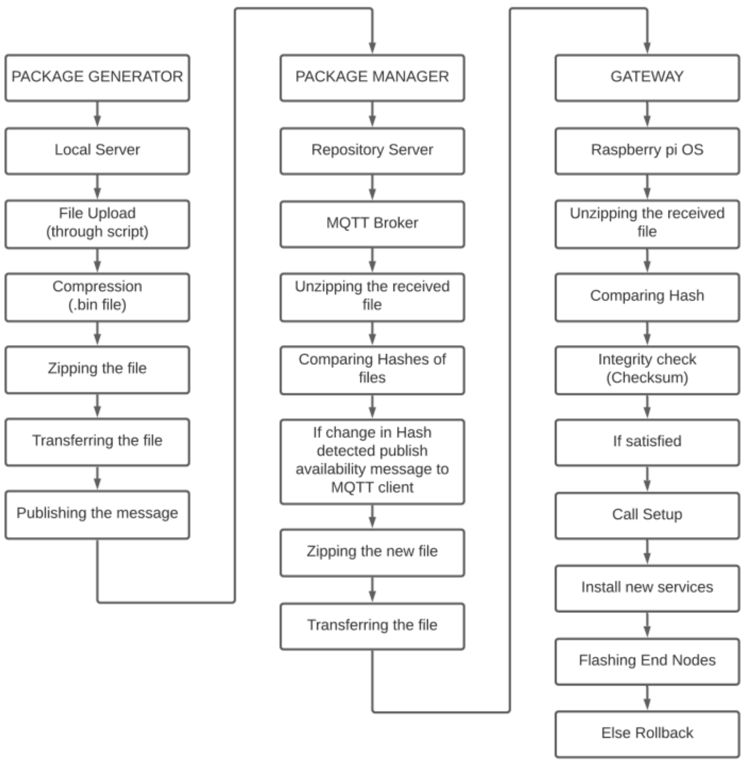


- Network architechture.
- The Network Architecture as shown in the fig. is designed based on the Elemental Model.
- In the Network Architecture diagram, the Firmware Patch Server i.e., Machine1 or the source machine will push the files to the Repository Server using port 22. The files are transferred using FTP protocol.
- On the Local Machine SFTP Server has been Setup in the Linux Environment and files are being transferred over SFTP Service.
- As we are working with the Private Network, we are using NAT for translating our private IP address to the public IP Address.



- Network address translation
- At the Repository Server, the MQTT Broker has been set up in the Linux Environment. On the Broker side, Virtual Domain Network is present. So, as on the Broker side, we have Private Network, therefore We've local private IP Address on the Repository Server (i.e., Source Address), A static public IP address on the Router and then a specific destination address.
- The NAT translates the Private IP Address, which is not unique globally to the Public IP Address and Vice versa. In other words, NAT helps to connect the Private and Public Network together, due to which network devices have access to the Internet.
- Here, the Port Forwarding has also been enabled, so that we can Direct our request from the Repository Server to our Local Server.
- The File transfer is taking place over the SFTP Service. So, port 22 is been used.
- We are using Paramiko Framework of Python for Secured File Transfer, to the Repository Server.
- Once the files are pushed from the Source Machine i.e., Local Server to the Repository Server, The Repository Server will publish the availability message to the SoC. In our case we are using Raspberry Pi.
- As Raspberry Pi has subscribed to the MQTT Broker for New Firmware Availability, it functions as MQTT Client and Receives availability message for new firmware files.
- Then as MQTT Client, it sends Acknowledgement message to the broker- in the. get format, for eventually downloading the files.
- Here, As MQTT service is been used, it is TCP/IP Protocol based and we'll use Port 1883.We are using SFTP for file transfer, so it will take place through Port 22.
- The Raspberry Pi, then updates the firmware at Sensor Nodes with the New Firmware. This updating of files will be done by auto running python script as System Service, which is the predefined process.

Design Flow



V. Design flow

- The files are been uploaded from Machine 1 (Local Server) to Machine 2 (Repository Server) through SFTP, and the protocol used here is FTP.
- The modules that are developed will provide the following functionality:
- Hashing of files, Hashing provides with the Encryption of file and helps to uniquely identify a file and Replacing Old files with the new one.
- Compression of files in a particular format.
- All the tasks i.e., The Hashing, Replacing and Compression of the files will take place at Machine 1 itself even before the files are transferred to Machine 2.
- After file transfer, compressed files will be present on Machine 2.
- Firstly Machine 2 will publish Availability message to Machine 3 using MQTT protocol.
- After publishing the message, MQTT Port will be closed and immediately after that, at the same time SFTP Port will be opened for file transfer i.e., First message will be published through the port 1883 and then the port will be closed, and after that Port 22 will be opened for file transfer.
- When the file is transferred to Machine 3 (Raspberry Pi), it will update the firmware at the Sensor Nodes with the New Firmware.
- To access the Mqtt functionalities through python program, we need to install the Mqtt Python libraries on Raspberry Pi.
- To read the received data in python program, we need python Mqtt library.
- The most popular Mqtt library for python is Paho-mqtt library.

Acknowledgment

It gives us great pleasure and satisfaction in presenting the final project report on 'Firmware Update of IoT Devices Over the Air Interface'. We would like to take this opportunity to thank our internal guide Prof.Nabegha Masood and external guide Mr. Ajit Sarnaik and Mr. Basavaraj Hooli.We are really grateful to them for their kind support. Their valuable suggestions were very helpful. We would like to thank Mr. Pravin Raut network admin at MESCOE and all those, who have directly or indirectly helped us for the completion of the work during this seminar.

References

[1]. IEEE Transcations on Parallel and Distributed Systems, vol. 8, no. 5, pp. 449.461, 2014.
[2]. L. Atzori, A. Iera and G. morabito, "The internet of things: A survey", Computer Networks, vol. 54, no. 15, pp. 2787 2805, 2010.
[3]. SAM R21 Xplained Pro User Guide, Atmel Co., California, US, 2016, pp. 1 30.