COMMUNICATION PROTOCOL FOR A MULTIPLE INTERFACE GATEWAY IN IoT SYSTEMS

Zsolt BALÓ, Marian ALEXANDRU

"Transilvania" University, Brașov, Romania (zsole992@gmail.com, marian.alexandru@unitbv.ro)

DOI: 10.19062/1842-9238.2017.15.1.3

Abstract: *The concept of the IoT (Internet of Things) has become a very hot topic in fields like networking and embedded electronics, thanks to the fast development in the different types of processing units which gave rise of powerful, very compact and highly efficient systems like: SOC's (System on Chip) and OBC's (On Board Computer). These systems can also offer networking functionalities, and together with the IPv6 and its myriad number of IP addresses, the IoT can become the part of every household. In this paper, a particular system will be presented, which could be used as a part of an IoT system.*

Keywords: *IoT, OBC, networking, smart home*

1. INTRODUCTION

The original architecture of the classical internet was created long before the idea of internet of things (IoT) would be conceptualized, which means that it is necessary to realize a different type of communication method, which can handle an exponentially increased data streams from diverse sources like sensors, actuators or other kind of electronic devices with the capability to transmit data in a given medium.

The information transmitted by these devices will also be really different from the type of data that is exchanged in the traditional internet in the sense that it is more redundant, with a slow variation of the measured values. The medium in which the data is transmitted, in the most cases will be "low fidelity", the attenuation and the interference may cause some loss of data, but in this case a big portion of data is insignificant and can be extrapolated by the correctly received values. [1]

Considering the waste list of protocols used in IoT (RFID, ZigBee, GPRS, GSM, Bluetooth, Wi-Fi, Ethernet, …)[5] for wired and wireless communication it is really important to have a system which can handle multiple protocols. Based on these multiple protocols, a generic protocol could be created.

In order the receive data in IoT systems, an efficient method is to use the MQTT (Message Queue Telemetry Transport) or a similar protocol which is based upon a publish-subscribe system. This protocol allows the end user to choose only the necessary data, by subscribing to the published information using a broker, which has the role to manage the coordination of subscriptions and data reception.[6] After the data had been collected from the various sources, it could be sent to a Cloud network. The Cloud network can provide a decentralized system, which is highly reliable and accessible from anywhere and anytime.[4]

In this paper, a simple protocol is presented, which will encapsulate the transmitted data into a specific data packet with additional information to realize a publish- subscribe system similar to MQTT. In order to make the system easier to integrate into Cloud systems, it will use the httpRest protocol, which is based on the HTTP protocol (GET, POST, DELETE,...), which will provide semantic information, to realize a customizable back-end application.[3]

2. THE GATEWAY SYSTEM AND THE COMMUNICATION PROTOCOL

The description and the functionalities of the system realized and tested by the authors is presented below. The system uses a specific protocol, implemented using the UDP (User Datagram Protocol) transfer protocol, which has the role to organize data and to give additional meta-information which will be useful for further data processing. Also, it realizes a dialog between the main components of the system. This protocol will run in the Application OSI (Open System Interconnection) layer which presumes an existing connection between the communicating devices. The software program used for this system was developed in Python.[2]

Three types of data packets are used within the protocol between the gateway and the devices, each having a different purpose and slightly different structure, as follows:

a) The simple data packet

This type of packet is used for simple data transfer only, between the devices and the gateway. The dialog in the case is unidirectional.

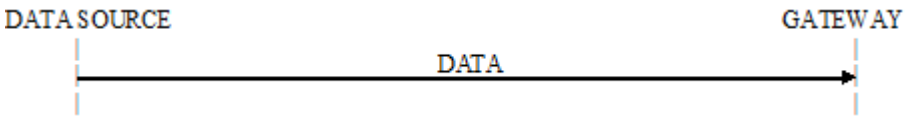

FIG. 1. Simple data packet

b) The management packet

The management packet is used to obtain status information from the device, if this kind of information is available, about the parameters measured by the device, in order to be able to respond with a corresponding control packet, to create a log file or a visual representation of the desired data.

When this type of packet is used, the dialog is bi-directional, first the gateway asking for the information using this type of packet, and, subsequently, if the device receives at least one packet, then it starts its own response stream. When the gateway receives the response, it means that the request was successful. Else, the streaming will continue for a given time value, after that the streaming being repeated. As an example, this packet is used to get information about the quality of the radio connection, the current channel, etc.

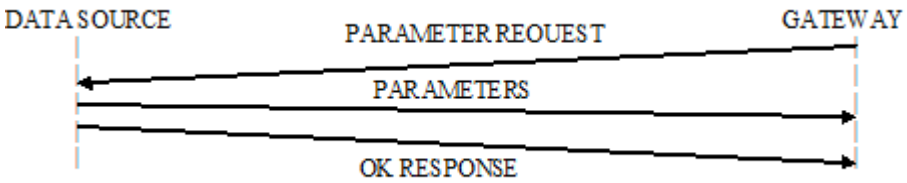

FIG. 2. Management data packet

c) The control packet

The purpose of this packet is to control the different kind of settings of the devices connected to the gateway, or to eventually stop the communication with the device, if necessary. After a stream of control packets the gateway will wait for a response packet from the device and in the case, that it doesn't had received any response it will retry the transmission for a given amount of time. If the communication is successful the gateway will receive a packet about the result of the command.

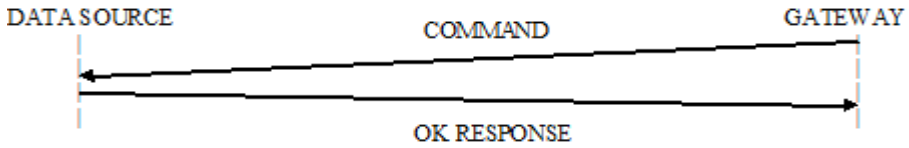

FIG. 3. Control data packet

This part of the system will use UDP or a similar protocol, for example L2CAP (Logical Link Control and Adaptation Protocol) in the case of Bluetooth communication (if different kind of communication modules like the SIM800L GPRS module is used the communication will be over UART - Universal Asynchronous Receiver/Transmitter, or other available communication interface, so in this particular case there is no need of continuous data streams or retransmissions).

The reason for this decision is that the data in the majority of cases has a really slow variation in time and the use of a protocol which is connection-oriented requires an excessive overhead, which is quite unnecessary in this case, because the same data will be transmitted over a long period.

The connectionless transmission has its drawbacks too, which in this case is the increase of the complexity of the communicating software, which requires a way to assure at least best-effort communications.

The second protocol which is used between the gateway and the main server is similar to the one described before, having the difference that it uses a connection over TCP (Transmission Control Protocol). The difference is that instead of continuous data streaming, it is enough to transmit a data only once.

2.1. The data packet structure. The packet type used by these communication, uses the structure illustrated below:

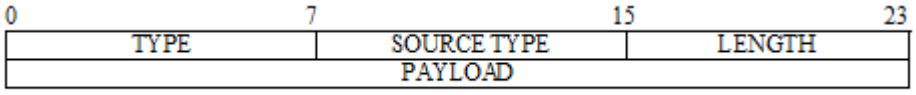

FIG. 4. Data packet

This structure contains two main parts: a header and a payload part, which is used to differentiate the communication packets used by the protocol. The header section consists of three parts each with its own use in the detection of: the used communication type, the source which have sent the data and the payload length (in the examples presented in the paper the payload length is 1 byte, because the source of the data which will be transmitted is an 8-bit microcontroller, and the length of the measured value is also 8 bits).

The PAYLOAD section contains the value which was measured in one side, and is transmitted to the other side. The length of this section can differ by the nature of the data transmitted, the data size used by the measuring unit, or the type of the packet. When different communication types are used, other than simple data communication, the payload content is the value that was requested by the transmitting end.

The TYPE section has a set of predefined values which the communicating parts can use to decide what kind of data had been sent or received (Command Packet, OK, Error, Management, …).

The SOURCE TYPE is used in order to decide from which interface did the data arrived and what kind of procedures can be used in the case of management or control communication.

The LENGTH part of the header section, is used by the receiving part to proceed in a way that all the data transmitted could be received without errors.

3. USE CASE SCENARIO OF THE SYSTEM

To demonstrate how the system works, a practical representation of the system will be shown below. From the devices listed above (RFID, ZigBee, GPRS, GSM, Bluetooth, Wi-Fi, Ethernet, …), the communication realized with the Wi-Fi device will be presented here. There will be three major parts in this section of the paper, which refers to the different kind of data packets which can appear in a conversation between the gateway and the device. Each of these cases will be monitored by the Wireshark program and presented, to put emphasis on the structure and the content of the data used in the respective communication.

Case 1. The simple data packet

This packet contains the value measured by a temperature sensor and send forward to the gateway for future processing and/or retransmission to the final server. To watch the variation of the incoming data a graphical user interface was added to the program in the gateway side, realized with the Python scripting language. In the picture below, the decoded information coming from the data source can be seen:

FIG. 5. The graphical interface of the program containing the received data

In order to decide the information type and source, the program uses the respective sections from the received datagram which can be seen in the following Wireshark capture:

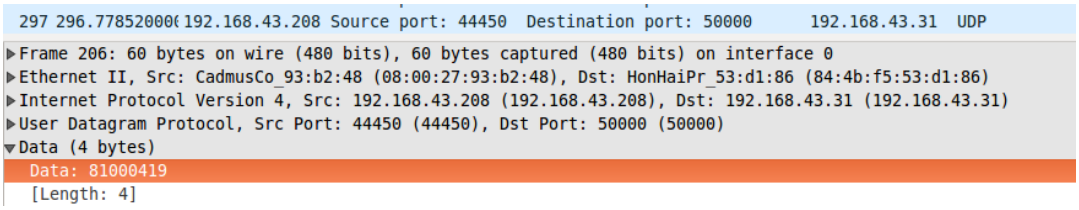

FIG. 6. Wireshark capture of the received data

In the Fig. 6. it can be seen that 40 bytes have been received, from which the first 5 are useful and the rest are used to fill the empty sections in order to have similar sizes with the management and control packets. The 5 bytes are 81.00.04.19 which corresponds to 1000.0001 – TEMPERATURE, 0000.0000 - WIFI, 0000.0100 – 4 bytes of useful data, and 0001.1001 – the value 25.

Case 2. The management packet

The management data packet is used to obtain information from the measuring side, in our case about the RF (Radio Frequency) channel parameters and the access data. After pressing the get parameters button the program will send a request to the data source and it will receive the requested data, and at last the response message, which stops the request stream.

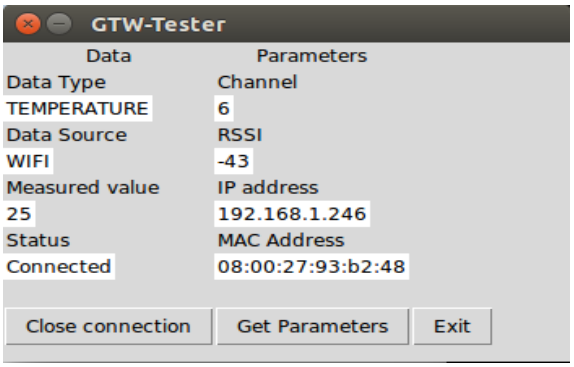

FIG. 7. Parameters received from the data source

The parameter request packet can be seen below and the first response packet also, which contains the requested parameters.

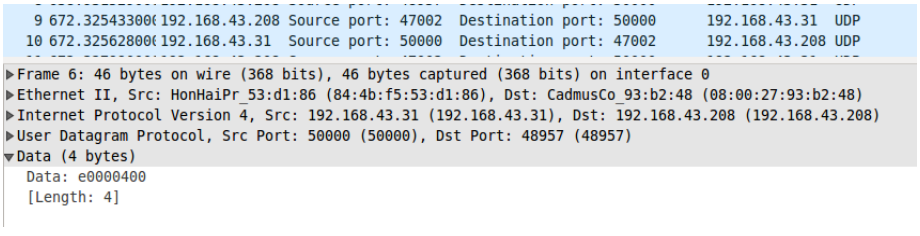

Fig. 8. Wireshark capture of the data request

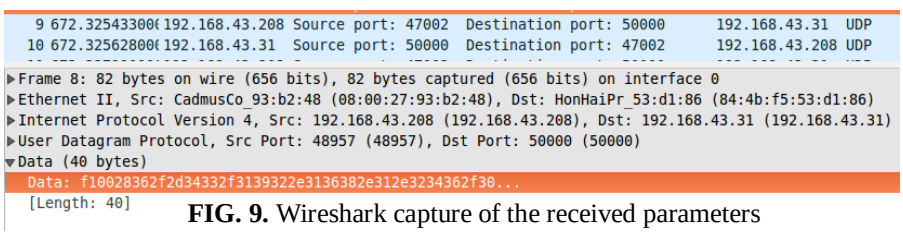

FIG. 9. Wireshark capture of the received parameters

Case 3. The control packet

The control packet is used to control the transmission from the measuring side and to obtain a response from the measuring side about the state of the execution of the command.

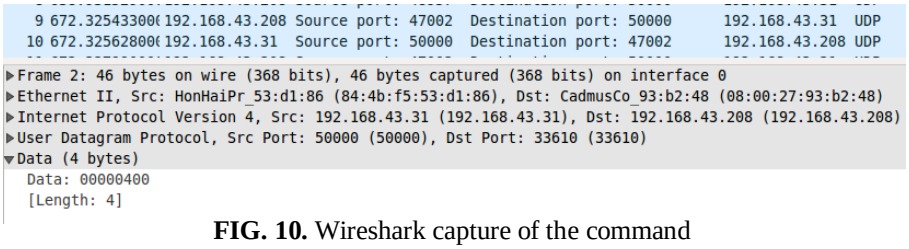

FIG. 10. Wireshark capture of the command

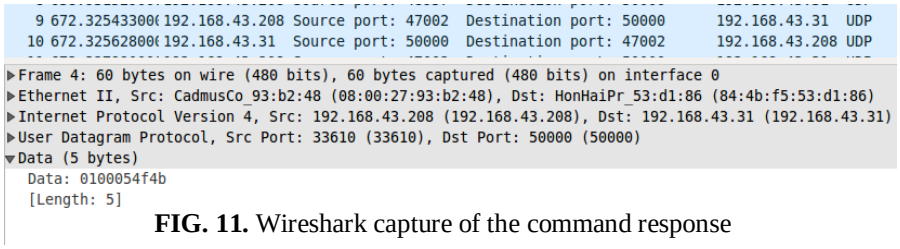

FIG. 11. Wireshark capture of the command response

In the captures from the Fig. 10 and 11 the sent command is presented with the response from the data source.

CONCLUSIONS AND FUTURE WORK

An example of a particular realization of a communication gateway which can be developed further and can be integrated inside a more complex system was presented in this paper.

For future development, it is planned to design a network interface with a server, making the role of a gateway similar to a broker used in MQTT systems, which can ask for the desired data, in a subscription manner, and also to transmit commands and parameter requests. These actions will be fulfilled using the same protocols which was presented in the previous chapters but in this case using the TCP and HTTPREST protocols, which can offer the security of the transmitted data.

REFERENCES

[1] F. daCosta, B. Henderson, *Rethinking the Internet of Things: A Scalable Approach to Connecting Everything*, ISBN 978-1-4302-5740-0, Apress, 2013;

[2] B. Rhodes, J. Goerzen, *Foundations of Python Network Programming 3rd ed*., ISBN 978-1430258544, Apress, 2014 Edition;

[3] T. Yashiro, S. Kobayashi, N. Koshizuka, K. Sakamura, *An Internet of Things (IoT) Architecture for Embedded Appliances*, Humanitarian Technology Conference (R10-HTC), 2013 IEEE Region 10, 26-29 August, http://ieeexplore.ieee.org/document/6669062/;

[4] Y. Benazzouz, C. Munilla, O. Günalp, M. Gallissot, L. Gürgen, *Sharing User IoT Devices in the Cloud*, Internet of Things (WF-IoT), 2014 IEEE World Forum on, 6-8 March, http://ieeexplore.ieee.org/document/6803193/;

[5] M. J. Kirtikumar, M. Shubham, R.M. Banakar, *Evolution of IoT in Smart Vehicles: An Overview*, 2015 International Conference on Green Computing and Internet of Things (ICGCIoT 2015) Proceedings, p. 804, http://toc.proceedings.com/29121webtoc.pdf;

[6] U. Hunkeler & H. Linh Truong, A. Stanford-Clark, *MQTT-S – A Publish/Subscribe Protocol For Wireless Sensor Networks*, IBM Zurich Research Laboratory, Switzerland, IBM UK Laboratories, Hursley, UK , http://www.ieeelcn.org/prior/LCN34/lcn34demos/lcn-demo2009_munari.pdf.