Original Research Paper

A Comparative Analysis of Network Management Protocols in IoT Applications

[1]Levi Costa Mota, [2]Edward David Moreno,
[2]Admilson Lima Ribeiro and [2]Ricardo J.P.B. Salgueiro

[1]*Federal Institute of Sergipe, Aracaju, Sergipe, Brazil*
[2]*Federal University of Sergipe, Aracaju, Sergipe, Brazil*

Article history
Received: 5-08-2018
Revised: 8-09-2018
Accepted: 22-09-2018

Corresponding Author:
Levi Costa Mota,
Federal Institute of Sergipe,
Aracaju, Sergipe, Brazil
Email: levi.mota@gmail.com

Abstract: The complexity and growth of new smart objects networks are generating a new demand for the maintenance of these devices, with the need to remotely monitor and control these devices without consuming significant resources. Analyzing the memory and electric power consumption of the protocols used in the management of these networks is a way to highlight the best protocol alternatives for this type of application. This study develops an experimental study, analyzing the behavior of SNMP, Zabbix and MQTT protocols, in terms of memory and electric power consumption, when it is used in an Internet of Things application, with a sensor device implemented on the ESP8266. The experiment is performed by monitoring devices in an environment with some Motes and a Zabbix server. The study analyzes the ROM and RAM memories occupied by the firmware code, in addition to the electric power consumption of each protocol. At the end, the study confirms that the three protocols analyzed are supported by the platform used. The research shows that the SNMP is the protocol that consumes the least amount of device memory, that there is no significant difference in the energy consumption between the protocols and that the MQTT protocol is suitable to be used in this environment and it also enables a significant reduction in energy consumption.

Keywords: Internet of Things, IoT, Management Protocol, SNMP, Zabbix, MQTT

Introduction

The Networked Embedded Systems (NES) are increasingly present in the current world, whether through Wireless Sensor Networks (WSN), or because of new smart objects that, by being interconnected and then connected to the worldwide network, have created the new concept of the Internet of Things (IoT).

It is estimated that, currently, each person has at least two objects connected to the Internet and that by 2020 this number may reach 20.8 billion worldwide. These smart objects are sensor devices with processing limitations, network range, battery capacity and amount of memory. They are generally used in applications with little or no human interaction and their networks are usually of large-scale, on the order of thousands of nodes.

In order to keep the NESs working properly, it is essential to apply management techniques that allow monitoring the functioning of the elements or send commands to change their behavior (Sheng *et al*., 2015a).

Numerous researches have been carried out to address the management of resource-limited device networks, both in a broader context of management (Sheng *et al*., 2015b) and specifically in monitoring the elements of these networks (Paventhan *et al*., 2013). However, there is no study that compares the protocols that can be used for network management, analyzing the memory and electric power consumption of the agents, in order to guarantee the applicability of these protocols in the management of devices with few resources.

The network management is usually performed by including management agents in the network elements. The agent is a software component that implements a management protocol and has the ability to interact with a server to provide the required management functionality. This component adds new capabilities to the device, but it also adds the resource consumption of the device that will be managed.

Given this scenario, there is a necessity to get to know the real impact of the network management protocols applied to the Internet of Things environment, taking the unique characteristics of this environment. It is necessary to know what resource consumption is added to the device after the inclusion of the management agent that implements the communication protocol with the management server.

This study analyzes the SNMP, Zabbix and MQTT network management protocols, with the objective of assessing the memory consumption and electric power consumption of the management agent, in the context of the IoT applications, in embedded systems implemented with the ESP8266.

The article is organized in 4 sections. Section 1 presents the introduction, the main related studies, the fundamental concepts related to management of NES and the protocols used in this work. Section 2 deals with the materials and methods, detailing what has been done for its development. Section 3 presents the results obtained in the tests performed, while section 4 offers the conclusions and suggestions for future studies.

Related Works

While performing the bibliographic review, no studies dealing specifically with the analysis and comparison of network management protocols of embedded systems were found. However, the following studies were found in the area related to the management and monitoring of NES and the Internet of things, including the analysis of the protocols used and suggested by the authors.

In "Design and Implementation of Embedded SNMP Network Management Manager in Web-Based Mode" (Wu *et al*., 2008), the authors present an architecture of network management of embedded systems based on SNMP and in Web mode in Linux. However, metric analysis are not done in order to assess the performance of the protocol in the proposed environment, nor are other protocols analyzed.

Mukhtar *et al.* (2008), in "LNMP- Management Architecture for IPv6 Based Low-Power Wireless Personal Area Networks (6LoWPAN)", proposes the LoWPAN Network Management Protocol (LNMP) as the management architecture for WSN based on 6LoWPAN. It is suggested the use of a gateway to separate the IPv6 and 6LoWPAN networks. The analysis performed by the authors is about the latency of the requests regarding of the number of jumps performed by the packages to reach the destination. They do not assess the memory consumption and do not compare with other protocols.

Kuryla and Schönwälder (2011) carried out a study to see if the SNMP protocol could be applied to devices with limited resources. They implemented a SNMP agent on the Contiki operating system and analyzed its behavior on an 8-bit Raven AVR platform. The analysis was performed in relation to the ROM and RAM used by the agent. Kuryla's method of analysis was the same method chosen for this work. However, Kuyla analyzed only the SNMP, not comparing it with other protocols.

The authors of "Management of Resource Constrained Devices in the Internet of Things" (Sehgal *et al*., 2012) have made an assessment of two management protocols applied to the internet of things environment. The authors assessed the SNMP protocol for monitoring and the NETCONF protocol applied to the device configuration. The metrics used are also based on ROM and RAM consumption. However, the authors only analyze the management aspects of a single protocol.

"Design and Implementation of a WiFi Sensor Device Management System" is a project proposal for a management system for sensor devices based on 802.11 networks (Cai *et al*., 2014). The authors proposed a sensor device management model based on light protocols such as the CoAP. An evaluation of the agent's memory usage is performed by assessing the firmware layers separately. The package sizes for each message type are also analyzed.

As shown above, all the studies evaluate only one protocol and one aspect of resource consumption. Some studies do not have an environment that is true to the current IoT scenario, or they do not use devices with limited resources. This study presents an analysis of three protocols applied to management in IoT, under the memory and power consumption aspects.

Management of Networked Embedded Systems

Historically, the computer networks have been marked by complexity, diversity and growth. These characteristics make it difficult to maintain these environments. In addition, as computer systems networks are becoming critical to modern business environments, monitoring and ensuring their reliability in performance is absolutely necessary. A way for keeping the computer networks running smoothly is to use solutions that allow the management of elements of those networks.

NES also have difficulties that are similar to those of common networks. In addition, the smart devices, which often have several resource limitations, add even more complexity to the task of managing those networks. In order to maintain these sensor devices monitoring performance or sending commands to the sensor node, for instance, it is essential that a communication protocol that is efficient and does not consume considerable resources is used (Sheng *et al*., 2015a).

Models and protocols have been developed for network management and the main exponent of these technologies is the SNMP protocol. But some other protocols can be availed for the same purpose.

SNMP Protocol

The Simple Network Management Protocol (SNMP) is a network management and monitoring protocol that runs on the UDP protocol (Paventhan *et al*., 2013). Its last specification is the RFC-1157. This is an application layer protocol of the TCP/IP stack. Its architectural model has a collection of network management stations and network elements, which are called managed devices. Each managed device contains asoftware called "agent", which is responsible for handling the manager's requests. It is possible to affirm that, currently, the SNMP is the standard protocol for managing TCP/IP networks (Salvador and Granville, 2008).

Zabbix Protocol

Zabbix is a modern, open source and multiplatform network management tool, with a distributed monitoring system capable of monitoring the availability and performance of a network infrastructure, as well as its applications.

In addition to the tool, Zabbix developers have also defined a protocol for communicating with equipment agents that do not support traditional protocols such as SNMP. The Zabbix protocol is extremely simple and works over TCP connections. The Zabbix agent, which implements the management protocol, is deployed to a monitoring target and then provides data for the local resources.

MQTT Protocol

The Message Queue Telemetry Transport (MQTT) is a messaging protocol introduced by Andy Stanford-Clark of IBM and Arlen Nipper of Arcom in 1999, and it was standardized in 2013 (Al-Fuqaha *et al*., 2015). Its primary purpose is to connect embedded devices to applications and middlewares. It works over the TCP protocol and uses the default publish/subscribe, in which clients do not have to request updates impulsively, thus reducing the depletion of the nodes resources (Katsikeas *et al*., 2017).

The MQTT protocol is not advertised as a network management protocol, but after a detailed analysis it is clear that its use in this type of application is perfectly possible, since telemetry technology allows things to be measured or monitored remotely. The MQTT is suitable for use in situations in which the network has low bandwidth or high latency and with devices that may have limited processing and memory capacity.

Materials and Methods

The environment used to perform the experiments and collect the data to be used in the assessment is shown in Fig. 1 and its operation is described below.

The Administrator uses any network station that has a Web browser. By using the browser, it is possible to access the Zabbix Web Interface. In this same virtual machine, the Zabbix server is also running, which accesses the Motes, made on the ESP8266, through the management protocols. In the Motes, the software component that responds to the consultations is the management agent. This is sufficient to meet the SNMP and Zabbix protocols but is not sufficient for the operation of the MQTT protocol.

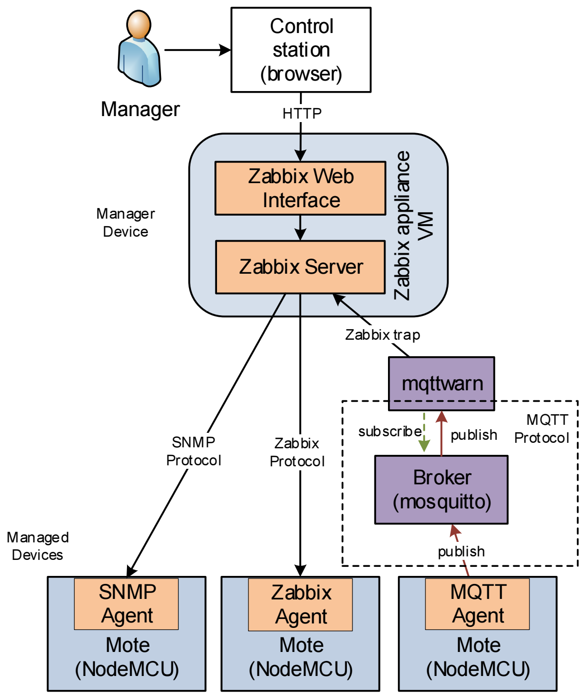

Fig. 1: Experimental environment components diagram

Manager Device

The manager device is an equipment in which the management server runs. In order to work as a manager device, Zabbix was installed and configured from its distribution in the form of an appliance-type virtual machine. It has facilitated the server and management interface operation process.

Managed Device

In order to play the role of the managed device a Mote was built on the ESP8266. The ESP8266 is a microcontroller that has been standing out as a low-cost alternative for the implementation of smart objects. The characteristics of this new System On a Chip (SOC), such as a 32-bit processor, RAM of about 100KB and 512K Flash memory, have been decisive for its use in several scientific researches carried out in the last two years (Kodali and Soratkal, 2016; Marques and Pitarma, 2016).

A comparative analysis was also made between ESP8266 and existing devices such as TMote Sky, AVR Raven, WisMote and Arduino BT, as can be seen in Table 1. In this comparison it is possible to observe the advantage of the ESP8266 in the "Processor", "RAM", "ROM" and "Wireless network standards".

A Mote is a device that brings with it some sensors and a power source. As the ESP8266 does not have these elements, it was necessary to build a device that had its own power and sensors in order to obtain information from the environment, thus simulating a Smart Object.

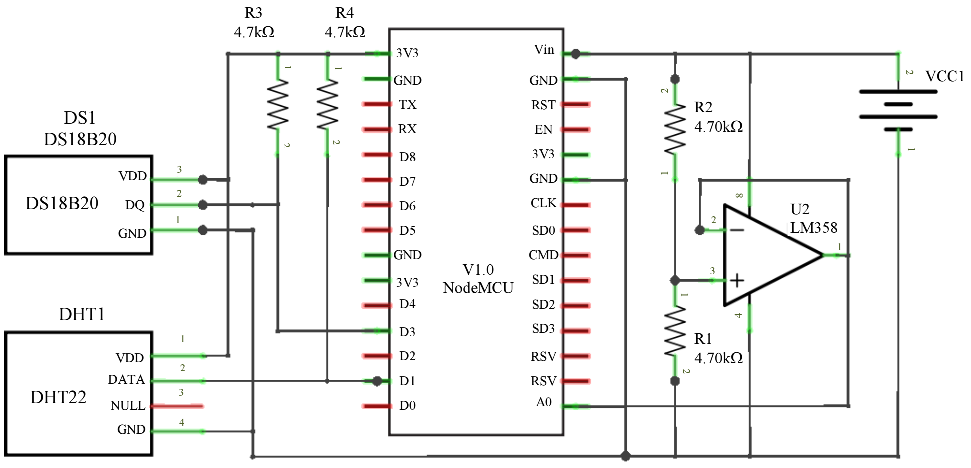

Fig. 2: Schematic diagram of the implemented device

Table 1: Use of ROM and RAM by the agents (in bytes)

Device	Processor	RAM (KB)	ROM (KB)	Network
ESP8266	Tensilica L106 32-bits RISC	32 + 80	512 Flash	802.11 b/g/n/e/i
Telos/ TMote Sky/ MTM-CM5000	Texas Instruments MSP430 F161116-bits RISC	10	48	802.15.4
AVR Raven	Atmel ATmega3290P + ATmega1284P 8-bits RISC	16	4	802.15.4
WiSMote	Texas Instruments MSP430 5 series 16-bits	16	128	802.15.4
Arduino BT	Atmel ATmega328 8-bits RISC	2	32 Flash	Bluetooth

In addition, it has been added to the device the ability to provide performance and failure measures, enabling a useful and effective remote management.

The schematic diagram of the device circuit is shown in Fig. 2. It uses as base a NodeMCU board, which includes the ESP8266. The circuit includes a DHT22 as a temperature and humidity sensor for the environment, a DS18B20 as the temperature sensor for the chip and to complete, a power supply meter made with a resistive tension divider and an operational amplifier configured as a tension follower.

Memory Consumption Metric

Considering that memory is one of the most limited resources of the devices that are used in IoT, the amount of RAM and ROM occupied by the agent are important parameters to be evaluated (Kuryla and Schönwälder, 2011).

The firmware stored on the device includes the code of the program that performs the function for which the device was developed, but also includes the agent code that allows its management remotely. The firmware code, including the agent, is recorded on a Flash-type ROM and this is loaded into RAM when it is run. The RAM stores, in addition to the code that will be executed, the constant and variable data used by the program. Thus, the less space of code and memory is occupied by the agent, the more room will be left for the main firmware code.

Power Consumption Metric

Power consumption is a great concern when it comes to embedded devices applied to IoT. Several researches focus on this aspect with the aim of saving the energy consumed by these restrict devices (Atzori *et al*., 2010; Feng *et al*., 2011; Dagale *et al*., 2015).

The concern regarding the power consumption is particularly important when the solution applies battery-powered devices running on field, without the use of the power grid. In these cases, the lower the power consumption, the longer the device's autonomy (Wang *et al*., 2006; Moui and Desprats, 2011).

Experiment Scenario

In the experiment scenario, the sensor device is managed and monitored by a management server, which collects the failure and performance data every ten seconds.

For convenience, every time the experiment is run, it lasts a total period of five minutes. During this period, one hundred and fifty requests are made from the server to the agent. As the server sends requests every ten seconds and at that time five different fault and performance information is requested, in the period of each test thirty bursts of five requisitions are obtained. This time is sufficient to confirm the dynamic memory consumption of the agents, whose stabilization already happens around three minutes of execution. In addition, during this time, it is possible to perform 16,100 samples of energy consumption of the device, enough to observe the linearity in the evolution of this consumption.

Results and Discussion

General Memory Consumption

The results of the ROM and static RAM collections of the three agents are presented in Table 2. As the collection was made by layers, it can be observed that the SDK layer occupies 221,995 bytes of ROM, which corresponds to 42.3% of the total and 31,568 bytes of RAM, corresponding to 38.5% of the total. The sum of the ROM of the other layers is 7.7%, 9.5% and 12.3% of the ROM occupied by the SDK in the SNMP, Zabbix and MQTT protocols, respectively. For the RAM, this comparison is 5.6%, 4.8% and 6.6%, in the same protocols.

ROM

The results of the ROM occupation by the agents are presented in Fig. 3. The lowest occupancy of ROM was that of the SNMP agent, followed by the Zabbix agent and, with the greatest occupation, the MQTT agent. It is possible to observe that the difference occurred due to the transport protocol used by the agents and due to the agent's own code. Regarding the transport protocol, the SNMP uses UDP messages and this is an advantage in terms of ROM when compared to Zabbix and MQTT, which uses TCP connections. It is worth remembering that the TCP protocol is connection-oriented (reliable), while the UDP does not have this concern. In addition, the UDP simplicity also results in much less code in the management agent.

Static RAM

The results of the occupation of static RAM are shown in Fig. 4. Based on total the occupied RAM, the Zabbix protocol performed better, occupying 248 bytes less than the SNMP firmware and 572 bytes less than the MQTT firmware. By analyzing the layers individually, it is clear the responsible for this was the Zabbix agent. The largest occupation of RAM was from the MQTT agent.

Evolution of Total RAM Consumption

The results of the collection of RAM occupied by the firmware over time can be seen in Fig. 5. The total RAM occupancy of the agents starts at near levels. This initial occupation is directly related to the static data used by the agent. The occupation of RAM of the SNMP agent keeps, during all execution, in the level of 35,224 bytes, except for just two moments in which this occupation quickly rises to 35,424 bytes. It demonstrates that the UDP messages exchanged over time little affect the occupancy of RAM.

The Zabbix agent experiences memory increase as it receives requests from the server. At the time of the establishment of each TCP connection, there is an increment of approximately 200 bytes of RAM, which is only released some time after the connection is finished. The ten-second interval between the requests is less than the disposal time of the connections, causing the memory to continue to increase. When reaching the threshold of 41,024 bytes, there is an equilibrium point. At this point, the memory allocation due to the initiation of new connections and the discarding of memory from previous connections keeps the memory occupancy at the same level until the end of the experiment.

The MQTT protocol starts the execution taking 35,728 bytes and remains at that level until the end of the experiment. It happens because the MQTT uses only a TCP connection and all the message exchanges happen over that connection. This takes up a memory space of about 200 bytes, but that remains stable throughout the execution of the agent.

Power Consumption

In order to reach the electric power consumption, the electric voltage and the electric current are also measured. The product of these last two quantities results in dissipated power. The energy consumption is the integral of powers over time.

The results of the power measurements over the run time of the experiment are shown in Fig. 6. Although power peaks of up to 1200 mW are observed, the average power dissipated remains in the range of 475 mW regardless of the protocol used.

Table 2: Use of ROM and RAM by the agents (in bytes)

	SNMP		Zabbix		MQTT	
Layer	*ROM*	*RAM*	*ROM*	*RAM*	*ROM*	*RAM*
SDK (SO)	221,995	31,568	221,995	31,568	221,995	31,568
Network and Data Link	1,018	76	1,018	76	1,018	76
UDP	2,256	128	0	0	0	0
TCP	0	0	3,208	192	3,208	192
Basic firmware	9,392	1,016	9,392	1,016	9,392	1,016
Agent	4,408	552	7,416	240	13,668	812
Total	239,069	33,340	243,029	33,092	249,281	33,664

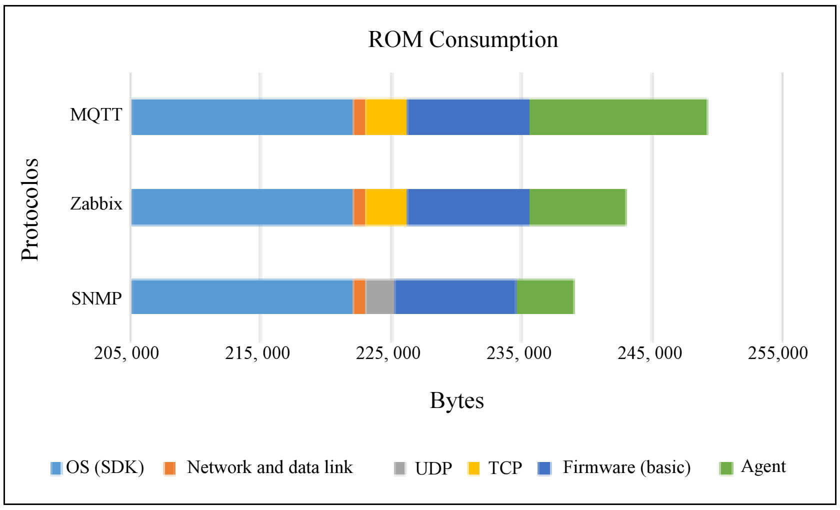

Fig. 3: Use of ROM by the agents

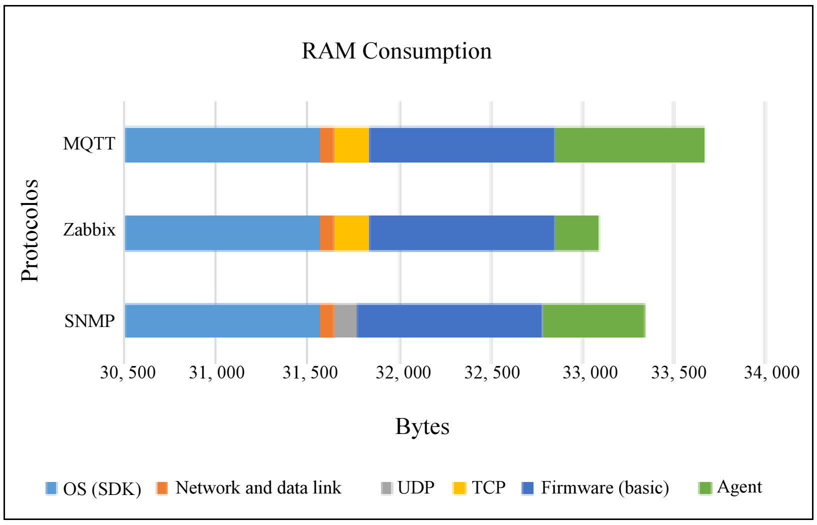

Fig. 4: Use of RAM by the agents

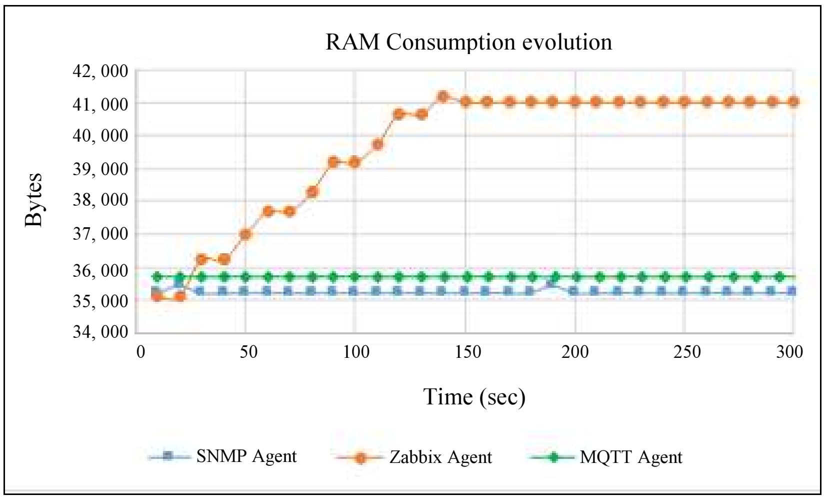

Fig. 5: Evolution of RAM consumption by the agents

Table 3: Power consumption of the agents

Firmware and agent	Average power dissipated (mW)	Total power consumed (mWh)	Estimated battery life of 4300mAh (hours)
SNMP agent	476.5	39.7	43.4
Zabbix agent	473.1	39.6	43.8
MQTT agent	475.0	39.6	43.9
MQTT (on sleep mode)	182.7	15.1	115.1

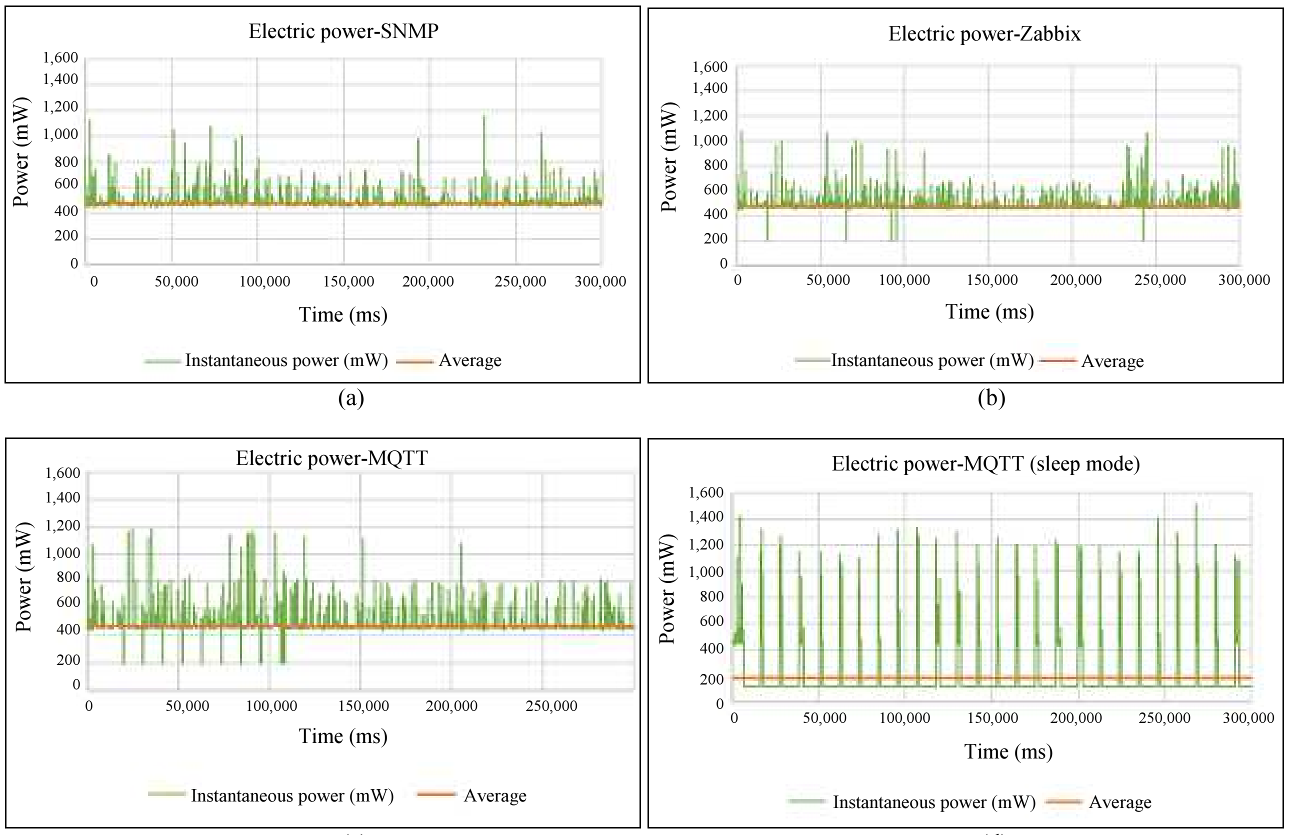

Fig. 6: Evolution of electric power

The experiment performed with the MQTT protocol, which places the device in suspend mode in the intervals between data collection and sending is a highlight. As can be seen in Fig. 6(d), the device starts its operation dissipating an average power of 475mW, however, after the first transmission of information and the device is placed in deep sleep mode, the dissipated power changes to a threshold of 120mW. When the device returns from sleep mode, the power has peaks of more than 1200mW returning to the 475mW threshold and starting a new cycle.

Table 3 shows the results of the measurements of the average power dissipated and it also details the total energy consumed in the five minutes of the experiments. In the table is also included a forecast of the 4300mAh battery life, if we consider what is consumed during the experiment. It is once more possible to observe that the great differential happens when the device is put into sleep mode. In this case, the average power dissipated during the entire execution of the experiment is 182.7mW, compared to the 475mW of the other cases. The total energy consumed in the experiment is 15.1mWh, against the average of 39.6% of the other cases. Moreover, the expected battery life goes from 43.7 hours to over 115 hours.

Conclusion

This study has pointed to the memory and power consumption of the main network management protocols implemented in agents installed on an ESP8266 device, when applied in an Internet of Things environment.

In addition, due to the analysis of the results obtained, it was possible to get to some more conclusions. The three protocols analyzed can be implemented and run on ESP8266 devices. The SNMP protocol agent performed best in terms of memory occupancy, both ROM and RAM and it is the best alternative among the three protocols. In many scenarios, the use of a standardized and consolidated protocol such as the SNMP may be desirable, or even required. In these cases, SNMP responds perfectly. The Zabbix protocol agent had the second best performance in ROM occupancy. Regarding the static RAM, it was better than the other two protocols. However, when analyzing the evolution of RAM occupation over the time of the experiment, this protocol was the worst among the three. It is possible to assume that this protocol may not withstand executions in scenarios with a higher request overload than the analyzed scenario, which had six requests every ten seconds for five minutes. The MQTT protocol agent, despite having occupied the most ROM and static RAM, has remained stable throughout the execution and can be a viable alternative if the application environment requires or prefers this recent protocol.

There is no significant difference in energy consumption between the protocols analyzed on the ESP8266 platform with a NodeMCU development board. However, the MQTT protocol agent, because of its characteristics, allowed the device to be put into sleep mode reducing by more than half the power consumption in that operating model.

Several future studies can be carried out seeking the continuity and improvement of this research, among them: the performance of a similar experiment, including more protocols in the analysis, such as Modbus, for instance; the analysis of the network traffic of the agents of each protocol; the analysis of the latency in the exchange of the messages between the device and the server; the application to other protocols, besides the MQTT, of the active agent management model, assessing its gains.

Acknowledgment

We would like to thank the Federal University of Sergipe and the Federal Institute of Sergipe for supporting this research.

Author's Contributions

All authors equally participated in all experiments, coordinated the data-analysis and contributed to the writing of the manuscript.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and there are no ethical issues involved.

References

Al-Fuqaha, A., M. Mohammadi, M. Aledhari and M. Ayyash, 2015. Internet of things: A survey on enabling technologies, protocols and applications. IEEE Communi. Surveys Tutorials, 17: 2347-2376. DOI: 10.5752/P.2316-9451.2013v1n2p78

Atzori, L., A. Iera and G. Morabito, 2010. The internet of things: A survey. Comput. Netw., 54: 2787-2805. DOI: 10.1016/j.comnet.2010.05.010

Cai, X., Y. Wang, X. Zhang and L. Luo, 2014. Design and implementation of a WiFi sensor device management system. 2014 IEEE World Forum Internet Things. DOI: 10.1109/WF-IoT.2014.6803108

Dagale, H., S.V.R. Anand, M. Hegde, N. Purohit and M.K. Supreeth *et al*., 2015. CyPhyS+: A reliable and managed cyber-physical system for old-age home healthcare over a 6LoWPAN using wearable motes. 2015 IEEE Int. Conference Services Comput. DOI:10.1109/SCC.2015.50

Feng, K., X. Huang and Z. Su, 2011. A network management architecture for 6LoWPAN network. Proceedings of the 4th IEEE International Conference on Broadband Network and Multimedia Technology, Oct. 28-30, IEEE Xplore Press, China, pp: 430-434. DOI: 10.1109/ICBNMT.2011.6155971

Katsikeas, S., K. Fysarakis, A. Miaoudakis, A. Van Bemten, I. Askoxylakis, I. Papaefstathiou and A. Plemenos. 2017. Lightweight & secure industrial IoT communications via the MQ telemetry transport protocol. Proceedings of the 2017 IEEE Symposium on Computers and Communications, Jul. 3-6, IEEE Xplore Press, Greece, pp: 1193-1200. DOI: 10.1109/ISCC.2017.8024687

Kodali, R.K. and S. Soratkal, 2016. MQTT based home automation system using ESP8266. Proceedings of the 2016 IEEE Region 10 Humanitarian Technology Conference, Dec. 21-23, IEEE Xplore press, India, pp: 1-5. DOI: 10.1109/R10-HTC.2016.7906845

Kuryla, S. and J. Schönwälder, 2011. Evaluation of the Resource Requirements of SNMP Agents on Constrained Devices. In: Managing the Dynamics of Networks and Services, Chrisment I., A. Couch, R. Badonnel and M. Waldburger (Eds.)., Springer, Berlin, Heidelberg, pp: 100-111.

Marques, G. and R. Pitarma, 2016. An indoor monitoring system for ambient assisted living based on internet of Things architecture. Int. J. Environ. Res. Public Health, 13: 1152. DOI: 10.3390/ijerph13111152.

Moui, A. and T. Desprats, 2011. Towards Self-Adaptive Monitoring Framework for Integrated Management. In: Managing the Dynamics of Networks and Services, Chrisment I., A. Couch, R. Badonnel and M. Waldburger, (Eds.). Springer, Berlin, Heidelberg, pp: 160-163.

Mukhtar, H., K. Kang-Myo, S.A. Chaudhry, A.H. Akbar and K. Ki-Hyung *et al*., 2008. LNMP-management architecture for IPv6 based low-power wireless Personal Area Networks (6LoWPAN). Proceedings of the IEEE Network Operations and Management Symposium NOMS 2008-2008, April, 7-11, IEEE Xplore press, Brazil, pp: 417-424. DOI: 10.1109/NOMS.2008.4575163

Paventhan, A., S. Krishna, H. Krishna, R. Kesavan and N. M. Ram, 2013. WSN monitoring for agriculture: Comparing SNMP and emerging CoAP approaches. Proceedings of the Texas Instruments India Educators' Conference 2013, April 4-6, IEEE Xplore press, India, pp: 353-358. DOI: 10.1109/TIIEC.2013.69

Salvador, E.M. and L.Z. Granville, 2008. Using visualization techniques for SNMP traffic analyses. Proceedings of the IEEE Symposium on Computers and Communications, July, 6-9, IEEE Xplore press, Morocco, pp: 806-811. DOI: 10.1109/ISCC.2008.4625672

Sehgal, A., V. Perelman, S. Kuryla, J. Schonwalder and O. In, 2012. Management of resource constrained devices in the internet of things. IEEE Communications Magazine, 50: 144-149. DOI: 10.1109/MCOM.2012.6384464

Sheng, Z., C. Mahapatra, C. Zhu and V.C.M. Leung, 2015a. Recent advances in industrial wireless sensor networks toward efficient management in IoT. IEEE Access, 3: 622-637. DOI: 10.1109/ACCESS.2015.2435000

Sheng, Z., H. Wang, C. Yin, X. Hu, S. Yang and V. Leung, 2015b. Lightweight Management of Resource-Constrained Sensor Devices in Internet of Things. IEEE Internet Things J., 2: 402-411. DOI: 10.1109/JIOT.2015.2419740

Wang, Q., Y. Zhu and L. Cheng, 2006. Reprogramming wireless sensor networks: Challenges and approaches. IEEE Network, 20: 48-55. DOI: 10.1109/MNET.2006.1637932

Wu, X., Y. Zhu and X. Deng, 2008. Design and implementation of embedded SNMP network management manager in web-based mode. Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference, Dec. 9-12, IEEE Xplore press, Taiwan, pp: 1512-1516. DOI: 10.1109/APSCC.2008.151