SNMP converter and the forward data collection as management method in dynamic distributed networks

Mauricio Tamayo[1] and Jorge Ortíz Triviño[2]

Universidad Nacional de Colombia, Faculty of Engineering,
Av Carrera 30 No 45 03, Bogotá, 111321, Colombia,
[1]`emtamayog@unal.edu.co`,[2]`jeortizt@unal.edu.co`

Abstract. A method of network management based on the protocol conversion SNMP to others, like serial or UDP, to manage small devices in a wireless sensor network is shown, whose reference framework combines the hierarchical network management with the dynamic, distributed and heterogeneous characteristics of an Ad Hoc network and how the SNMP protocol accomplish its habitual devices management functions, and at the same time, some of its messages are used to build a forward data collector that expands and contracts dynamically the storage capacity of the SNMP tables of the agent, adapting to the amount of devices in the network.

Keywords: Network management, SNMP, protocol conversion, finite-state converter, Ad Hoc, distributed processes, forward data collection

1 Introduction

This work is part the research project TLÖN, where a computational scheme inspired in social models is proposed, considering the justice, immanence, paradigm, government, existence and essence concepts [1] and supported in a multiagent system [2] which requires a management system. The results of the implementation of a protocol converter to include devices with administration restriction [3] like small sensors, are shown in this paper. In the section 2, it explains briefly the methods to address some challenges in the network administration in Ad Hoc networks. In the section 3, the SNMP architecture is analyzed to understand the protocol and use it to build the protocol converter. In the section 4, it gives the protocol design, how some challenges are faced, the message mapping and the forward data collection concept. Finally, part of the results and conclusions of the implementation are detailed.

2 Management systems in Ad Hoc networks

The researches about management systems for Ad Hoc networks are centered in the protocol development as ANMP [4], GUERRILLA [5] or LiveNCM [6], which are trying to mitigate the impact of the inherent to the operation of these networks as the energy limited usage, the wireless medium restrictions, the nodes heterogeneity, the autoconfiguration and automanagent. In addition, the continuous technology and scientific develops give tools to respond the mentioned challenges in turn new efforts and concepts appear like the use of a mathematical architecture named STEPS (Step Rate Storage) to model the collaborative management storage and data rate control in Wireless Sensor Networks (WSN) [7], the blockchain usage to develop distributed management systems in MANET [8] (Mobile Ad Hoc Networks), or the design of architectures based on SDN (Software Defined Networks) y NFV (Network Function Virtualization) to monitor the UAVs (Unmanned Aerial Vehicles) telemetries [9].

When we found such varied works about the network management in a lot of environments, it means that information is relevant and vital to the network operation and the services it provides.

In general, the sensors are configured through serial connections, where they can be managed and programmed to capture and show the sensed information. If you want to obtain such information through a management protocol such as those mentioned above, we will find an incompatibility of protocols. LAM [10] mentions that this situation can be seen as a problem of interoperability of processes and could be solved using common protocol images and designing a protocol converter of finite state machines.

In this way and considering the challenges of managing small devices with management limitations, that is, they do not support management protocols, and processing restriction, it is proposed to use protocol conversion by combining a finite state converter with the functions of a proxy agent SNMP through serial ports [3] and extending its use to wireless interfaces.

This work is not focus to evaluate various management protocols or investigate if SNMP is the ideal protocol for managing the WSN. We are exploring for a method of administration through a standard protocol in telecommunications (SNMP has emerged as the most widely used and deployed network management framework [11]), integrating the in Ad Hoc network management with other existed services or systems, perhaps in hierarchical topologies.

3 SNMP Architecture

SNMP is an application protocol by means of which the variables defined in the MIB (management information base) of the managed device can be inspected or altered [12]. The SNMP-based management systems use the client-server paradigm, composed of a manager, an agent, managed resources and SNMP messages [13].

The implementation of the SNMP architecture in its third version is called SNMP entity, which consists of an engine and one or more associated applications. The engine provides all the services of sending and receiving messages through the dispatcher, with the processing model supports and treats messages in all its versions, executes authentication and coding with the security model and access control to the managed objects [14] The SNMP applications, which are defined in RFC3413 [15], make use of the services provided by the engine. They include the command generator (monitor and manipulate management data), command responder (provides access to management data), originator of notifications (initiates asynchronous messages), receiver of notifications (processes asynchronous messages) and proxy forwarder (forwarding of messages between entities) [13]. In the figure 1, the described architecture is shown and taken into account for the solution proposed.

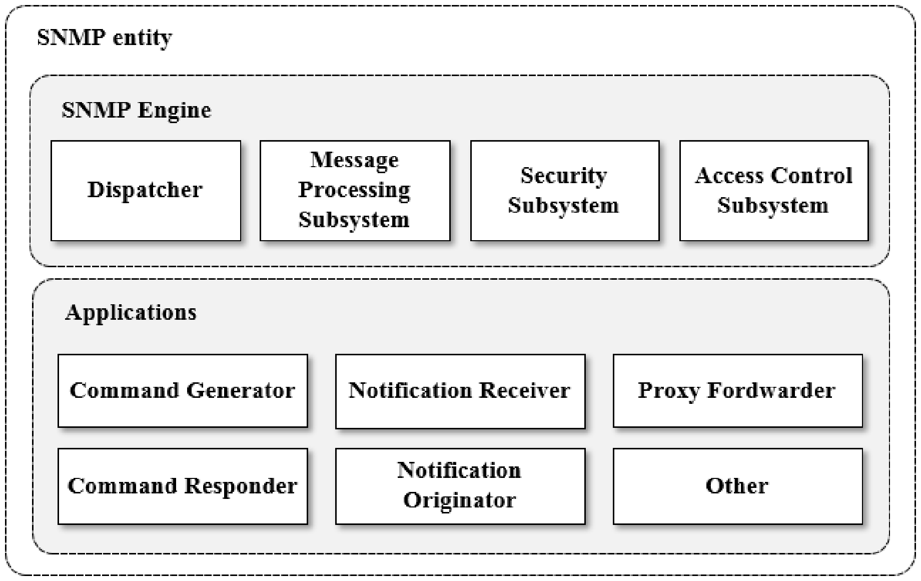


Fig. 1. SNMP entity architecture. Source: [14]

3.1 Proxy SNMP

In general terms, a SNMP proxy refers to an application for forwarding SNMP messages without taking into account which managed objects are contained within those messages [15], for example, forward SNMP requests from one transport domain to another, or translate them from one version to another. However, this solution is a little closer to another definition that is given in that same RFC that is the translation of SNMP requests in operations of some non-SNMP management protocol.

4 Protocol Designed

Based on the protocol conversion model using finite state machines [10], an operation scheme was designed so that non-SNMP devices could be managed. Finite state machines of each protocol were built independently using the service primitives that are necessary from each one to leave their protocol image. Then a relational finite state machine was integrated that establishes the communication process between the two protocols. The final state machine that represents the flow of messages and their states is shown in the figure 2.

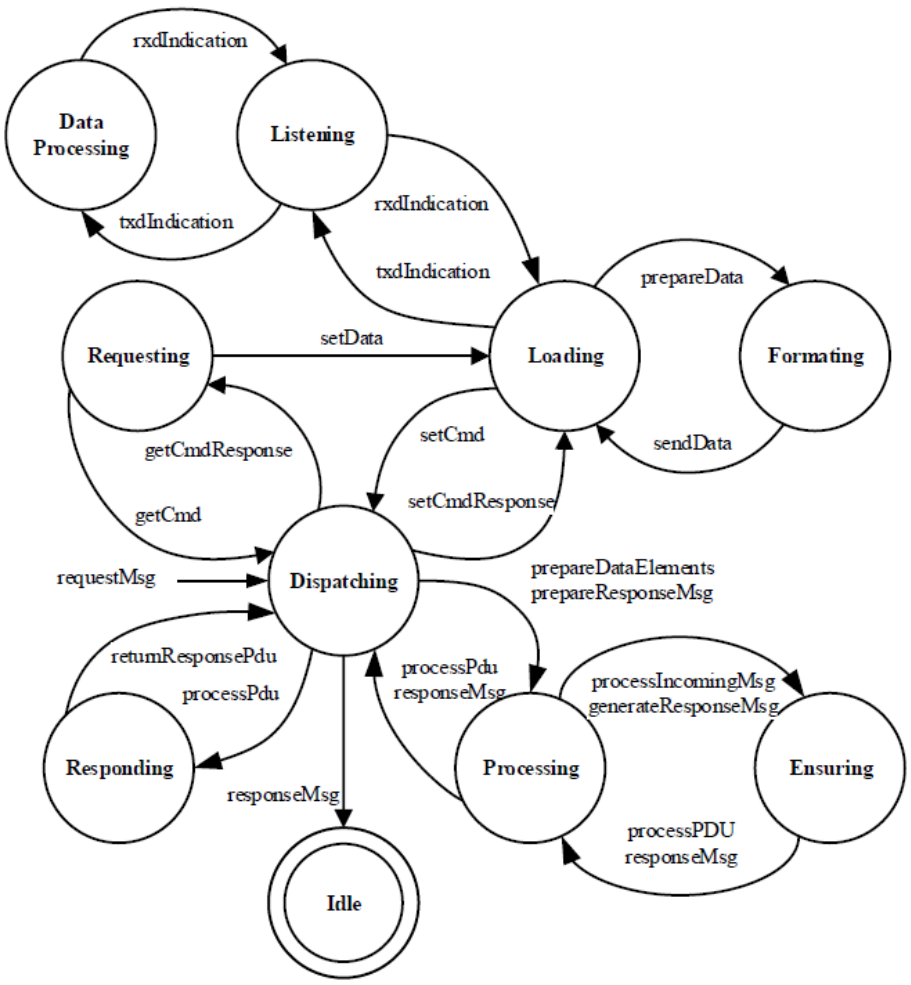


Fig. 2. Finite state machine. Source: Own

4.1 Challenges

Based on the experience during the development of the application, we consider that the main challenges for a SNMP converter protocol (or even pure SNMP) to be used in an Ad Hoc network are heterogeneity, dynamism and distributed operations. Next, we explain how each of these challenges was addressed, which finally constitute the basis of the protocol architecture proposed.

Heterogeneity Network management through SNMP requires the definition of objects in the MIB management information base. This base is designed and built knowing the device, its functions and the things that can be measured. In this sense and when a network is heterogeneous, the complicated issue is not to manage a great diversity of devices, the difficult thing is to cover the equipment management that are not known since they are not included in the MIB. One of the characteristics of Ad Hoc networks is self-configuration, so creating and/or modifying the MIBs autonomously is ideal, but it is a challenge that becomes, by itself, a rather complex job that is out of reach from this project.

The construction of the MIB was done employing the MIB Smithy software[1] whose design follows some recommendations of Walsh [16] like it must be syntactically correct, without mixing SMIv1 and SMIv2, validating the MIB in various compilers, etc. Thereby, with each object MIB, which are scalars, a conceptual table is created, using a tabular structure as mentioned in RFC2578 [17], generating an ordered collection of objects. For example, the OID 1.3.6.1.4.1.49843.1.1.1.2.1[2] is a table that represents a sensor that measures pressure and temperature like the BMP180 (see the MIB tree in the figure 6). Each object is configured as a table based on the dynamism of Ad Hoc networks.

Dynamism WSNs undergo dynamic topology changes due to the entry or exit of a node from the network, although compared to MANETs these changes are made to a lesser extent [18]. To make dynamic connections through SNMP, the use of fixed OIDs for each object is omitted, since the number of hosts that are going to operate within the network is not known, while it is important to manage the reuse of physical and logical resources. For this reason tables are configured that have conceptual rows that allow the creation or elimination of instances of objects [17] through a columnar object called RowStatus whose value represents the state of its row in the SNMP table. There are six possible values that define this state: active (1), notInService (2), notReady (3), createAndGo (4), createAndWait (5), destroy (6) [19]. The value of RowStatus can be changed through setRequest commands, either with a string value or an integer value.

When a new device enters the network, an available row is searched for to be assigned though a setRequest command, modifying the value of RowStatus with

[1] http://www.muonics.com/Products/MIBSmithy/

[2] The object identification tree OID has numbering assigned by the IANA to the Universidad Nacional de Colombia. https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers

4 or createAndGo. With this new row, a new OID is assigned adding a branch in its structure, e.g. the sensor BMP180 has the OID 1.3.6.1.4.1.49843.1.1.1.2.1, a new device of this type will be assigned to 1.3.6.1.4.1.49843.1 .1.1.2.1.1 (add a branch) and if additional sensor is connected to the network, it will have the OID 1.3.6.1.4.1.49843.1.1.1.2.1.2. When the device leaves the network, the row is deleted placing the value 6 or destroyed in the RowStatus field, freeing the resource that can be used with new accesses to the network.

Distribution Usually SNMP runs in centralized networks under the client-server model. In this work a part of this concept is kept in relation to the manager's queries to the agent and storage of the MIB. However, all processes of info request, data storage in the base of information management and detection of input/output of devices in the network, are done splitting them in multiple instances that can be executed in different devices with their own local resources interconnected into the network as a distributed computing system [20].

4.2 Message mapping

Since the conversion of protocols can be seen as a problem of interoperability of processes [10], the protocol design considers each protocol with its corresponded messages as an independent process that communicates with an intermediate entity that processes them independently. This design allows the execution of each process in a distributed way and permits to increase the range of protocols that could be converted to SNMP. In this work, the protocols serial and UDP were tested. An example is displayed in the figure 3, the process J (SNMP) sends messages to I (conversor), which treats the packets, in case it has the information to answer, it will send the respective message to the source or in other cases a message is forward to the S process (serial) and when it sends back the responses, I puts the information in the MIB to answer to J. In this instance, the converter has SNMP agent functions to response SNMP requests or generating events and notifications, and a DTE (data terminal equipment) device generating serial communication commands.

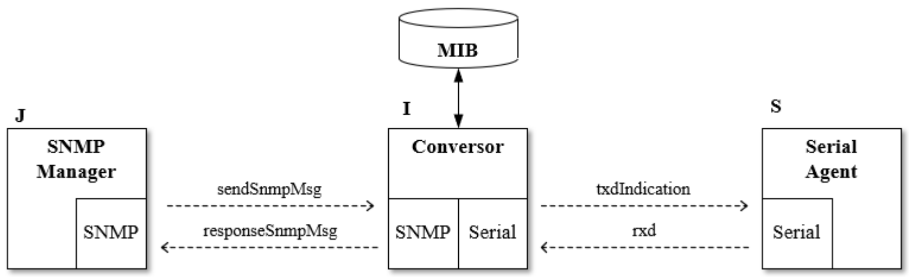


Fig. 3. Interoperability of JIS processes. Source: Own

4.3 Architecture

There is a SNMP agent, which contains the necessary software to build the base SNMP tables, defined in the MIB. In addition, as RFC1157 [12] states that it is mandatory that all SNMP implementations support five PDUs: GetRequest, GetNextRequest, GetResponse, SetRequest, and Trap (or Notification), the agent has the functions of command responder and notifications originator. There is another process that is responsible for communication against non-SNMP devices, using their medium, protocol and specific format (serializing the information to send it through a TTY port or creating a UDP socket for wireless communications) to execute information requests whose answers they are collected and formatted and then stored in the SNMP tables of the agent using setRequest commands. Thus, when getRequest messages are sent from the SNMP manager to the agent, this one already has the management information and responds in a typical SNMP process. The following figure shows the architecture of the protocol, whose modules were programmed with Python language using already existing libraries such as pysnmp [21] and pyserial [22] , both facilitate the developments in SNMP services and serial communications respectively.

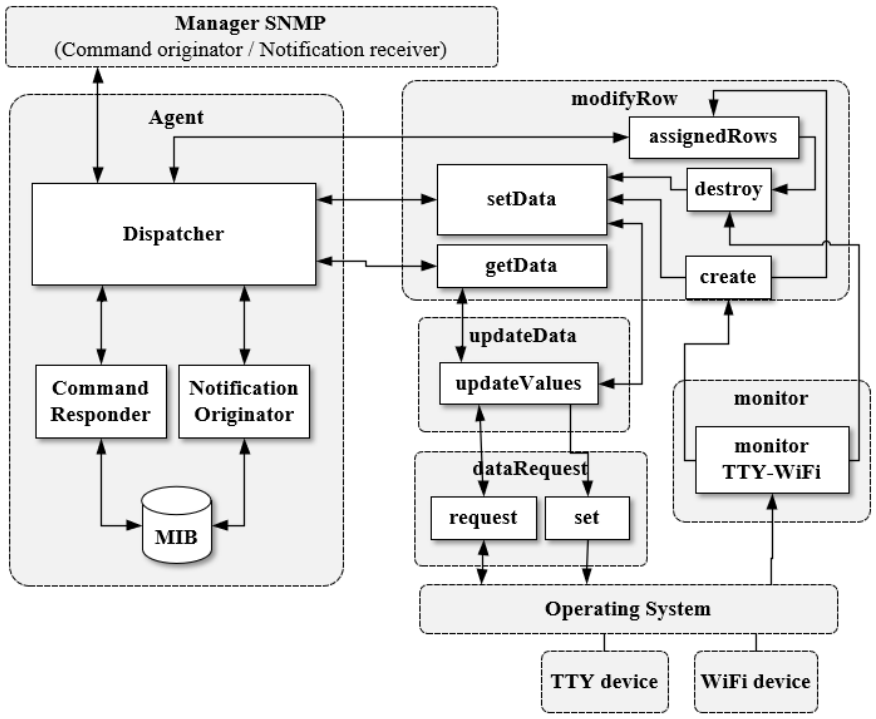


Fig. 4. Arquitecture of the protocol converter SNMP to serial or UDP. Source: Own

Forward data collection The automatic detection of a new device generates a new row within the table according to the type of device. A responsible process of validating the hosts registered in the tables, makes periodic request of each management object in each device, updating the fields in the table to maintain its consistency with the management information. So when the SNMP manager makes a query through a specific OID, the agent will be able to respond immediately. This process is named forward data collection (FDC), the stored information may or may not be used, but the process allows to increase the effectiveness in the delivery of the management information as well as its response time. Although there is some generation of traffic to obtain the information previously, this modality allows the development of distributed management processes and automatic detection of topology changes (input/output of host in the network) that could be executed by different devices within of the Ad Hoc network, for example one device can take the role of responder agent and another can be the forward collector, and in case the first mentioned leaves the network, another can assume its functions, even the same collector.

4.4 Implementation

The implementation of the protocol was done on a small network of sensors trying to cover different types of physical connections, virtualization, topologies and formats. This implementation design allow for the environment where the protocol is going to work and three items were considered: operating system, application and protocol-specific execution [23]. The laboratory topology used is shown in the figure 5. Two Raspberry Pi B3 with Raspbian operating system were used to create an Ad Hoc network, where the modules shown in the architecture (figure 4) were charged in each device to operate in a distributed way. One of them was working as agent SNMP and the another one simulated various sensors for testing UDP protocol. Additionally, an Arduino UNO with ultrasound proximity sensor HC-SR04 was connected via USB to one of the Rasperrys to test wired serial communication. Despite the project covers Ad Hoc networks, a BMP180 temperature and pressure sensor connected to a Nodemcu ESP8266 was also used with a WiFi connection in infrastructure mode. An external machine runs a SNMP network management application, fulfilling the functions of manager.

The tests included some habitual SNMP operations like get subtree information which generates a get request message for each object created in the MIB and its results are displayed in the figure 6. The devices and their objects were created in the manager software to monitor periodically the sensors mentioned above, one of the historical graph is shown int the figure 7. While the manager sensed the devices, the exchange packets were capture with a a sniffer (figure 8).

The checking procedure gave good results in both topologies, Ad Hoc and Infrastructure, and the SNMP manager was able to capture the data from all devices with their correspond connections and protocols. The monitor service detected topology changes creating a row in the SNMP table when the device access to the network and destroying when the device leaves the network.

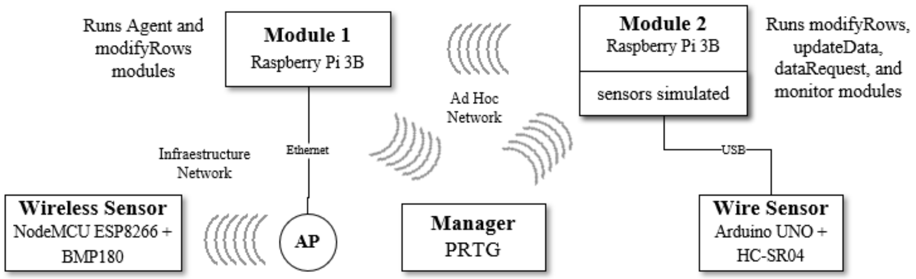
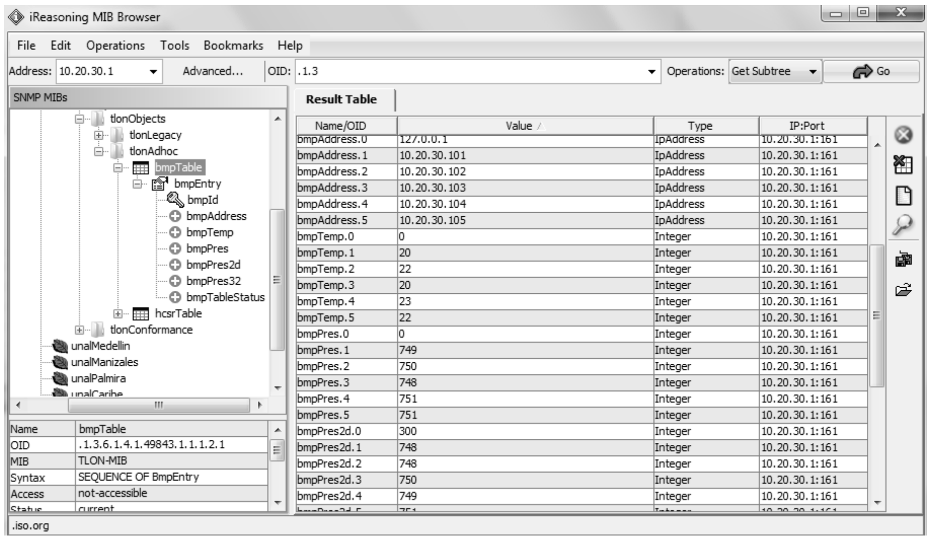


Fig. 5. Laboratory implemented to test the protocol designed. Source: Own



Name	bmpTable
OID	.1.3.6.1.4.1.49843.1.1.1.2.1
MIB	TLON-MIB
Syntax	SEQUENCE OF BmpEntry
Access	not-accessible
Status	current

Name/OID	Value	Type	IP:Port
bmpAddress.0	127.0.0.1	IpAddress	10.20.30.1:161
bmpAddress.1	10.20.30.101	IpAddress	10.20.30.1:161
bmpAddress.2	10.20.30.102	IpAddress	10.20.30.1:161
bmpAddress.3	10.20.30.103	IpAddress	10.20.30.1:161
bmpAddress.4	10.20.30.104	IpAddress	10.20.30.1:161
bmpAddress.5	10.20.30.105	IpAddress	10.20.30.1:161
bmpTemp.0	0	Integer	10.20.30.1:161
bmpTemp.1	20	Integer	10.20.30.1:161
bmpTemp.2	22	Integer	10.20.30.1:161
bmpTemp.3	20	Integer	10.20.30.1:161
bmpTemp.4	23	Integer	10.20.30.1:161
bmpTemp.5	22	Integer	10.20.30.1:161
bmpPres.0	0	Integer	10.20.30.1:161
bmpPres.1	749	Integer	10.20.30.1:161
bmpPres.2	750	Integer	10.20.30.1:161
bmpPres.3	748	Integer	10.20.30.1:161
bmpPres.4	751	Integer	10.20.30.1:161
bmpPres.5	751	Integer	10.20.30.1:161
bmpPres2d.0	300	Integer	10.20.30.1:161
bmpPres2d.1	748	Integer	10.20.30.1:161
bmpPres2d.2	748	Integer	10.20.30.1:161
bmpPres2d.3	750	Integer	10.20.30.1:161
bmpPres2d.4	749	Integer	10.20.30.1:161

Fig. 6. Results of get subtree command: Own

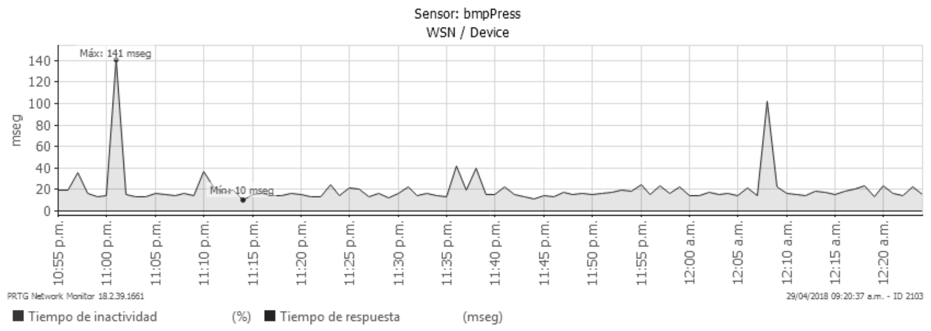
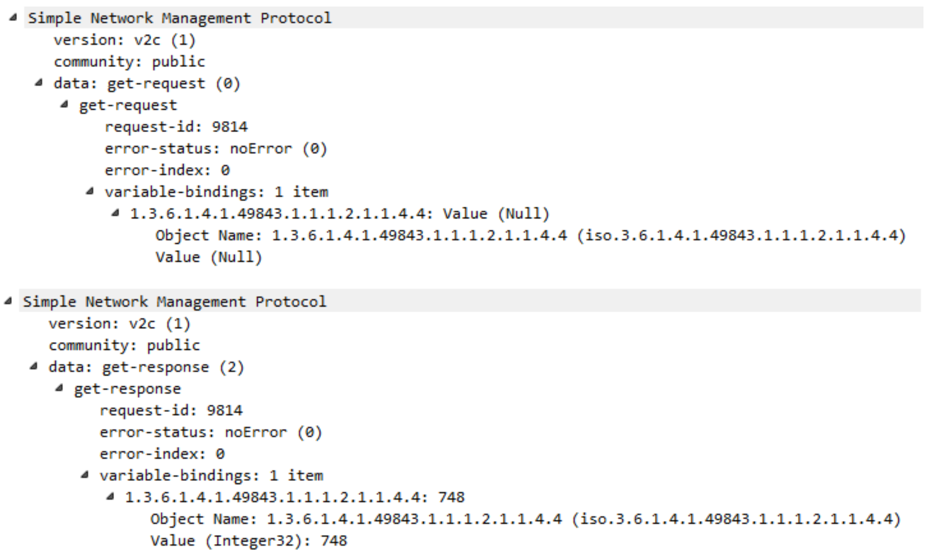


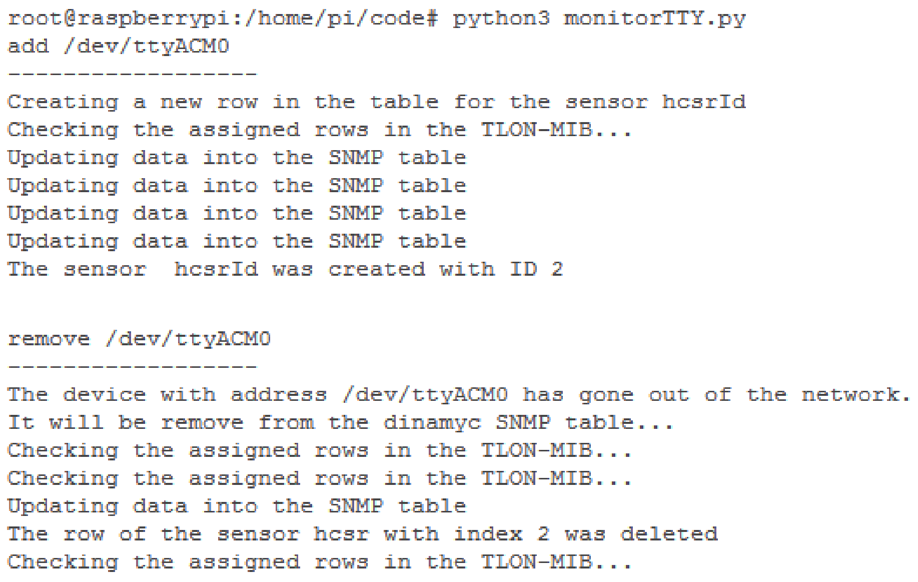
Fig. 7. Historical graph of a sensor Source: Own



```
◢ Simple Network Management Protocol
    version: v2c (1)
    community: public
  ◢ data: get-request (0)
    ◢ get-request
        request-id: 9814
        error-status: noError (0)
        error-index: 0
      ◢ variable-bindings: 1 item
        ◢ 1.3.6.1.4.1.49843.1.1.1.2.1.1.4.4: Value (Null)
            Object Name: 1.3.6.1.4.1.49843.1.1.1.2.1.1.4.4 (iso.3.6.1.4.1.49843.1.1.1.2.1.1.4.4)
            Value (Null)

◢ Simple Network Management Protocol
    version: v2c (1)
    community: public
  ◢ data: get-response (2)
    ◢ get-response
        request-id: 9814
        error-status: noError (0)
        error-index: 0
      ◢ variable-bindings: 1 item
        ◢ 1.3.6.1.4.1.49843.1.1.1.2.1.1.4.4: 748
            Object Name: 1.3.6.1.4.1.49843.1.1.1.2.1.1.4.4 (iso.3.6.1.4.1.49843.1.1.1.2.1.1.4.4)
            Value (Integer32): 748
```

Fig. 8. Get and get-response packets capture with sniffer. Source: Own



```
root@raspberrypi:/home/pi/code# python3 monitorTTY.py
add /dev/ttyACM0
------------------
Creating a new row in the table for the sensor hcsrId
Checking the assigned rows in the TLON-MIB...
Updating data into the SNMP table
Updating data into the SNMP table
Updating data into the SNMP table
Updating data into the SNMP table
The sensor  hcsrId was created with ID 2

remove /dev/ttyACM0
------------------
The device with address /dev/ttyACM0 has gone out of the network.
It will be remove from the dinamyc SNMP table...
Checking the assigned rows in the TLON-MIB...
Checking the assigned rows in the TLON-MIB...
Updating data into the SNMP table
The row of the sensor hcsr with index 2 was deleted
Checking the assigned rows in the TLON-MIB...
```

Fig. 9. Console messages when a device enters or leaves the network. Source: Own

5 Conclusions and future work

The service architecture designed permits the conversion from multi-protocols to SNMP, because the messages exchange between them is not done directly. The concept of FDC is a method that uses the MIB tables as main data base of the service, so the integration with another data bases schemes is not necessary, in addition, this idea permits the SNMP operations faster and minimize the lost data. To divide the conversion process in various modules facilitates the implementation in distributed networks. Finally, no matter if it is an Ad Hoc or infrastructure network or what kind of routing protocol is used, or even the device that is trying to manage, the proposed method for capture the data is executed successful, because it works in the application level and it does not intervene in the establishment or maintenance of the connexion between hosts and it only needs to include the code to get the data in the specific format of the device

Some challenges are pending to face like the auto-configuration of the MIB (the creation or destruction of MIB objects according to the kind of devices in the network), the mobility of the packets between the host to reassign the different functions in the architecture, to create new kind of data types through the TEXTUAL-CONVENTION (for example the feelings of an agent) for associating the basic concepts of a social model in the multi-agent system and to implement the solution using containers.

Acknowledgments. We appreciate the contributions of the research group TLÖN's members from Universidad Nacional de Colombia as well as Michael Kirkham from Muonics, Inc. who gave us a free academical license of MIB Smithy software to develop the MIB used in this project

References

1. TLÖN - Grupo de Investigación en Redes de Telecomunicaciones Dinámicas y Lenguajes de Programación Distribuidos and Universidad Nacional de Colombia. Proyecto TLÖN. http://www.tlon.unal.edu.co, 2017.
2. Henry Zárate-Ceballos, Joaquin Fernando Sanchez-Cifuentes, Juan Pablo Ospina-López, and Jorge Eduardo Ortiz-Triviñio. 44. Sistema de Telecomunicaciones Social-Inspirado mediante Comunidades de Agentes. *Cicom*, page 1, 2015.
3. Mauricio Tamayo Garcia, Henry Zarate, and Jorge Ortiz Triviño. Protocol Conversion Approach to Include Devices with Administration Restriction on a Framework of Reference of Management Network. *Communications in Computer and Information Science, vol 742.*, 742, 2017.
4. Wenli Chen, Nitin Jain, and Suresh Singh. ANMP: Ad hoc network management protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):1506–1531, 1999.
5. Chien-Chung Shen, Chaiporn Jaikaeo, Chavalit Srisathapornphat, and Zhuochuan Huang. The Guerrilla Management Architecture for Ad hoc Networks. *Proc. of {IEEE} MILCOM*, pages 1–6, 2002.

6. Aurelien Jacquot, Jean-Pierre Chanet, Kun Mean Hou, Gil De Sousa, and Antoine Monier. A new management method for wireless sensor networks. *2010 The 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pages 1–8, 2010.
7. A.H. Kabashi and J.M.H. Elmirghani. Adaptive rate control &amp; collaborative storage management for challenged ad hoc &amp; sensor networks employing static &amp; dynamic heterogeneity. *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, (3), 2010.
8. Sonoko Goka and Hiroshi Shigeno. Distributed management system for trust and reward in mobile ad hoc networks. *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6, 2018.
9. Kyle J S White, Dimitrios P Pezaros, and Matt D Knudson. A Programmable Resilient High-Mobility SDN + NFV Architecture for UAV Telemetry Monitoring. (June):2–7, 2016.
10. Simon S. Lam. Protocol Conversion. *IEEE Transactions on Software Engineering*, 14(3):353–362, 1988.
11. James F. Kurose and Keith W. Ross. *Computer networking : a top-down approach*. Pearson, sixth edition, 2013.
12. J Case, J Davin, M Fedor, and M Schoffstall. RFC 1157 A Simple Network Management Protocol (SNMP), 1990.
13. Mark A. Miller. *Managing Internetworks with SNMP*. Wiley, Foster City, CA, third edit edition, 1999.
14. D Harrington, R Presuhn, and B Wijnen. RFC 3411 An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, 2002.
15. D Levi, P Meyer, and B Stewart. RFC 3413 Simple Network Management Protocol (SNMP) Applications, 2002.
16. Larry Walsh. *SNMP MIB Handbook*. Wyndham Press, second edition, 2008.
17. K. McCloghrie, D. Perkins, J. Case, M. Rose, and S. Waldbusser. RFC2578 Structure of Management Information Version 2 (SMIv2), 1999.
18. Driss Benhaddou and Ala Al-Fuqaha. *Wireless Sensor and Mobile Ad-Hoc Networks*. Springer US, 2015.
19. K. McCloghrie, D. Perkins, J. Case, M. Rose, and S. Waldbusser. RFC2579 Textual Conventions for SMIv2 Status, 1999.
20. Pradeep K. Sinha. Fundamentals. In *Distributed Operating Systems:Concepts and Design*, volume 1, chapter 1, page 764. Wiley-IEEE Press, 1 edition, 1997.
21. Ilya Etingof. Python SNMP library for Python. http://pysnmp.sourceforge.net/, 2017.
22. Chris Liechti. Welcome to pySerials documentation. https://pythonhosted.org/pyserial/, 2015.
23. Hartmut König. *Protocol Engineering*. Springer-Verlag Berlin Heidelberg, Cottbus, Germany, first edition, 2012.