Research Article / Systems-Computer Engineering https://doi.org/10.15446/ing.investig.v42n1.87564

Design and Implementation of Network Monitoring System for Campus Infrastructure Using Software Agents

Diseño e implementación de un sistema de monitoreo de red para infraestructura de campus usando agentes de software

Rodrigo I. Espinel-Villalobos[1], Erick Ardila-Triana[2], Henry Zarate-Ceballos[3], and Jorge E. Ortiz-Triviño[4]

ABSTRACT

In network management and monitoring systems, or Network Management Stations (NMS), the Simple Network monitoring Protocol (SNMP) is normally used, with which it is possible to obtain information on the behavior, the values of the variables, and the status of the network architecture. network. However, for large corporate networks, the protocol can present latency in data collection and processing, thus making real-time monitoring difficult. This article proposes a multi-agent system based on layers, with three types of agents. This includes the collector agent, which uses a Management Information Base (MIB) value to collect information from the network equipment, an input table of information from the network devices for the consolidator agent to process the collected data and leave it in a consumable format, and its subsequent representation by the application agent as a web service, in this case, as a heat map.

Keywords: distributed systems, multi-agent system, network monitoring, parallelization, SNMP

RESUMEN

En los sistemas de administración y monitoreo de redes o *Network Management Stations* (NMS), normalmente se utiliza el protocolo *Simple Network Monitoring Protocol* (SNMP), con el cual es posible obtener información sobre el comportamiento, los valores de las variables y el estado de la arquitectura de red. Sin embargo, para las grandes redes corporativas, el protocolo puede presentar latencia en la recopilación y el procesamiento de datos, lo que dificulta el monitoreo en tiempo real. Este artículo propone un sistema multi-agente basado en capas con tres tipos de agentes. Esto incluye el agente recolector que utiliza un valor MIB(Management Information Base) para recolectar información de los equipos de red, una tabla de entrada de información de los dispositivos de red para que el agente consolidador realice el procesamiento de los datos recolectada y los deje en un formato consumible y su subsiguiente representación por parte del agente de aplicación como un servicio web, en este caso como un mapa de calor.

Palabras clave: sistemas distribuidos, sistema multi-agente, monitoreo de redes, paralelización, SNMP

Received: May 29th, 2020
Accepted: June 2nd, 2021

Introduction

Today's networks are dynamic, and numerous transactions per second between different clients, applications, sensors, and devices are needed to deploy new services, which consumes the information generated throughout the network ecosystem. Traditional communication between network devices (Core and End Users) generates transactions in traffic control. In most cases, the Simple Network Monitoring Protocol (SNMP) is used to analyze traffic, with the purpose of allowing the administrators to change and monitor the state of devices that support the protocol, for instance, by shutting down an interface, checking the users on a Vlan, counting the users' sessions, etc. (Stallings, 1998; Gonçalves *et al.*, 2012). The SNMP has two types of entities: managers and agents. Managers work in Network Management Stations (NMS) and receive messages and traps from SNMP agents. SNMP agents, in turn, provide management information to the NMS. The agent is software-executed on a network device and incorporated into the operative system by the user-administrator. The SNMP protocol uses a hierarchical structure called Management Information Base (MIB). This structure is a database of managed objects containing information about the devices and the network. The MIB has two types of objects: scalar

[1]Systems Engineer, Universidad Nacional de Colombia, Colombia. M.Sc. in Cybersecurity, INSA Centre Val de Loire. Affiliation: Researcher, Universidad Nacional de Colombia, Colombia. E-mail: riespinelv@unal.edu.co
[2]Systems Engineer, Universidad Nacional de Colombia, Colombia. M.Sc. in Telecommunications Engineering, Universidad Nacional de Colombia, Colombia. Affiliation: Chief of Communications Networks, Universidad Nacional de Colombia, Colombia. E-mail: eardilat@unal.edu.co
[3]Electronic Engineer, Universidad Central, Colombia. Ph.D. in Computer Science, Universidad Nacional de Colombia, Colombia. Affiliation: Researcher, Universidad Nacional de Colombia, Colombia. E-mail: hzaratec@unal.edu.co
[4]Systems Engineer, Universidad Nacional de Colombia, Colombia. Ph.D. in Computer Science, Universidad Nacional de Colombia, Colombia. Affiliation: Associate Professor, Universidad Nacional de Colombia, Colombia.
E-mail: jeortizt@unal.edu.co

How to cite: Espinel, R., Ardila-Triana, E., Zárate, H., and Ortiz, J. E. (2022). Design and Implementation of Network Monitoring System for Campus Infrastructure Using Software Agents. *Ingeniería e Investigación, 42*(1), e87564. https://doi.org/10.15446/ing.investig.v42n1.87564

and tabular. These are useful to present the NMS information. The Object Identifier (OID) is organized in a tree-like hierarchy as a set of integers separated by dots, with the purpose of instantiating objects with a unique ID, for instance, devices and their function (router, switch), interface name, interface state, and so on Kaushik, 2010). SNMP's modularity helps it evolve through three major versions and find widespread use and acceptance. The IETF recognizes SNMP version 3, defined by RFC 3411 and RFC 3418, as the current standard version of SNMP. The IETF has designated SNMPv3 as a full Internet standard, the highest level of maturity for an RFC. In practice, SNMP implementations often support multiple versions: typically, SNMPv1, SNMPv2c, and SNMPv3. SNMP is a useful and versatile protocol that can work with other network information data sets, neural networks, machine learning systems, multi agent systems, or other monitoring platforms. For instance, in Sánchez *et al*. (2013) machine learning uses clustering and visualization techniques to identify and locate anomalous SNMP situations, and thus get an idea of network performance. Other approaches are focused on information security and cybersecurity to detect network attacks based on machine learning and event correlation (Hwoij *et al.*, 2020; Al-Naymat *et al.*, 2019).

Our approach is a campus network testbed based on a multi-agent system with mobile agents to monitor and present the location of users on a campus network via the SNMP protocol. The proposed model has three types of agents, *i*) the collector agent, *ii*) the consolidator agent, and *iii*) the painter agent.

This paper is organized as follows: the Methodology section shows the process used to design and implement the model; the Related Work section presents a review of existing solutions similar to our proposal; the Framework Architecture section introduces the reference models adopted in the debate for the system architecture and describes its components; the Use case section explains the implementation and its scope; the Results section describes the implemented prototype and the experimental demonstration of the proposed solution, showing its effectiveness; and, finally, the conclusions are presented in the Conclusions section, along with future work.

Methodology

Based on previous applications used on the university campus to measure the number of users per building, a new model is proposed that incorporates a multi-agent architecture to optimize data collection times on the network. A multi-agent model with SNMP was designed, developed, and implemented to allow comparative measurements between the two systems, as well as to verify the change in collection response times. The procedure is classified in three stages: (I) a description of the multi-agent architecture and the different layers and agents of the proposed model; (II) a description of the implementation and scope of the model; (III) an analysis of results and proposals for improvement.

Related Work

System monitoring and detection through the SNMP protocol with *snmpwalk* command routines are a solution to monitor and collect information about the network. However, this approach has huge response times, and it decreases the device's performance by consuming computing resources such as CPU and memory while querying on the MIB. The model presented by Brattstorm and Morreale (2017) uses metrics, collection, storage, presentation, and alerting to notify the system or device administrator or central control process of abnormal system or device behavior, where presentation is the way of displaying control data and issuing alerts. Another approach is proposed by Affandi *et al*. (2015) with a network mapping tool that sends early warnings via Short Message Service (SMS). The process starts with metwork mapping using the SNMP agent on all network interfaces. The information contained in the agents is retrieved and processed as Transport Control Protocol (TCP) traffic. Finally, a monitoring system checks TCP traffic and the use of computing resources. This process allows the bandwidth and resources optimization to manage the emergency systems and control some variables via SNMP agents.

An implementation to monitor networks and systems is proposed by some authors with a generic matrix grammar (Min, 2011; García *et al.*, 2014). This framework uses some protocols to display and monitor the system, such as Windows Management Instrumentation (WMI), Computer Integrated Manufacturing (CIM), and SNMP to remotely collect and manage NMS data. The authors propose a monitoring automation engine with a matrix analyzer into the multi-agent-based solution. Additionally, the most widely used monitoring information systems use a traditional *snmpwalk* command to enable system integration with any management tool with SNMP support such as Nagios (Barth, 2008), SNMP MIB Browser Android Tool (Hidalgo and Gamess, 2014), and Manage Engine MIB Browser Free Tool (Free SNMP Walk Tool for Windows and Linux, n.d.). A layered approach is introduced by Wan *et al*. In this paper, a smart emergency system is proposed. This model collects data from urban areas, and its most important task is to detect an emergency and handle it in a timely fashion to ensure protect human integrity and the continuous operation of critic infrastructures.

Other auhtors have developed SNMP and multi-agent systems (Barruiso *et al.*, 2017; Moreira *et al.*, 2016; Torre and Yucelen, 2018). The first model is a proposal for a mechanism to provide flexibility and dynamism using virtual agent organizations (VO) embedded in different wireless sensors, which are embedded in devices with limited computational resources. The PANGEA framework incorporates features to the wireless sensor network such as adaptability, reorganization and learning capabilities, its agents have a control role in atomatically collecting and processing information. This approach allows using the MAS as a virtual organization to define the structure, roles, and hierarchies, thus regulating the interaction and communications between agents and network devices via SNMP and MIB analysis. The second approach uses the SNMP protocol to manage ubiquitous

systems through behavior graphs and a metric called the expected behavior representation (GoES). This method allows the link between SNMP agents and commercial off-the-shelf systems (COTS). The approach consists of characterizing the context of devices and testing whether they are able to understand what other systems are doing by monitoring their state. Finally, the approach with mobile agents described by Madi and Alkasassech, 2019) is a simulation with a Omnet ++ network simulator that aims to use the mobile agent on the network with different amounts of nodes (5, 10, 15, 20, and 30 nodes in the proposed scenario). The objective is to probe an algorithm to improve SNMP queries using two type of agents: link and data agents. The link agent is responsible for discovering the network and storing the connected nodes in the home server. The data agent is responsible for data retrieval.

Another approach focused on the choice of a methodology to design and implement multi-agent systems to design the monitoring system, as shown in Lin and Jiang (2014), to introduce the design steps and organizational methods such as GAIA and Ingenia. Based on the methodological approach, Isaza *et al.* used JADE as a multi-agent framework and the GAIA methodology to monitor 32 personal computers.

There are models without the SNMP protocol to monitor university campus networks with specific architectures, as is the case of Iot, deployed as a platform to characterize energy consumption (Moura *et al.*, 2021) in order to manage electrical resources or electrical grids (Chiadone *et al.*, 2019). Another approach used to monitor networks includes Software Defined Networks (SDN) techniques to monitor traffic flows on Iot devices. These models need two psychic channels to monitor the network: the data plane and the control plane (Pashamoktari *et al.*, 2020).

Finally, network monitoring has evolved into the field of cybersecurity as a monitoring system on a campus and industry infrastructure to identify and mitigate potential vulnerabilities (Laštovička *et al.*, 2020), in addition to analyzing the network flow with recurring neural networks (Yang *et al.*, 2020), wireless analysis (Jin *et al.*, 2019; Allahham and Rahman, 2018), or risk assessment (Awang *et al.*, 2020).

Our approach is an intermediate model proposed to monitor campus network core devices (routers, switches, and manageable network devices) and as a possible platform to deploy some monitoring systems for other variables taken from sensors, personal computers, or other linked devices within the campus network.

Framework Architecture

Our approach contemplates a three-layer architecture as shown in Figure 1 with mobile agents and a dispersion strategy to scan the campus network. Each of these layers has a specific task to accomplish an objective and finally obtain the number of users present in the network as a heat map. All the services are deployed on an virtual server on the private cloud infrastructure of the University.

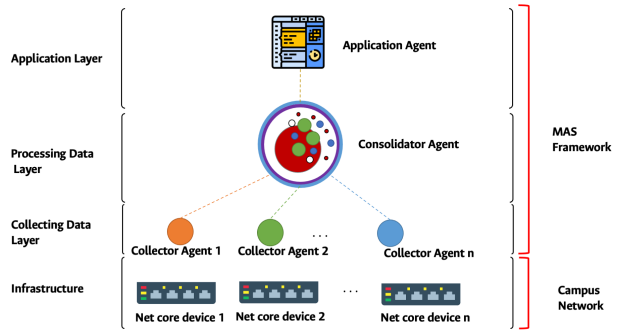

Figure 1. Proposed multi-agent system architecture.
Source: Authors

Data collection layer: It is the bottom layer that establishes communication with the SNMP agents using the Pysnmp library to make the query with the OID code on the MIB. It contains the collector agents (mobile agent).

Data processing layer: It is the intermediate layer that consolidates all the data obtained from all the collecting agents. It is also tasked with deploying and organizing the collecting agents. It summarizes, analyzes, and draws statistics from the data obtained.

Application layer: Its is the top layer that displays the collected information. It contains the application agent that creates the different web files to present data in the form of a heat map. Furthermore, all the data obtained from the preceding layers are also available in tables for ease of use and analysis.

PySNMP library

The library that has been used for our implementation is PySNMP (PySNMP 4.4.12, n.d), a free and open source implementation of the v1/v2c/v3 SNMP engine that is distributed under a 2-clause BSD license. It manages different layers as an evolution of SNMP We have used only the first layer of Pysnm2 in the implementation (Figure 2). This layer covers a basic and low-level protocol scope (SNMP v1/v2) in order to make a simple implementation and use all the power of the SNMP API to use the core functions of the SNMP protocol. Figure 2 shows the modular architecture of the library. Not all functionalities have been used in this work, but it can provide a handful of easy to use tools.

A brief description of the library is presented below:

- **SNMP engine:** it is a central object that controls the other components of the SNMP system.
- **Transport subsystem:** it is used to send SNMP messages and accept them from the network. The I/O subsystem consists of an abstract dispatcher and one or more abstract transport classes.
- **Message and PDU dispatcher:** its main responsibilities include dispatching PDUs from SNMP applications through various subsystems, all the way down to the transport dispatcher, as well as passing SNMP messages from the network up to SNMP applications.

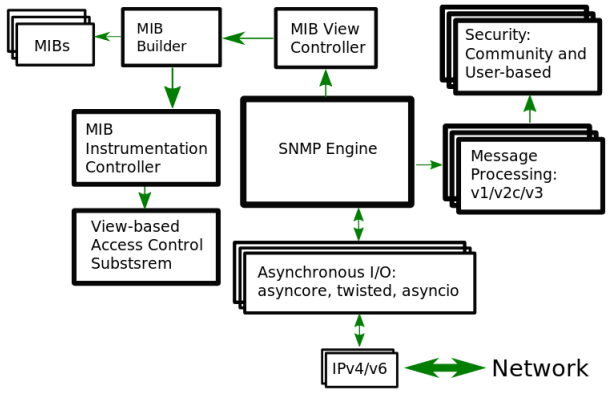

Figure 2. PySNMP library.
Source: Authors

- **Message Security Modules:** it performs message authentication and/or encryption. At the time of this writing, the User-Based (for v3) and Community (for v1/2c) modules are implemented in PySNMP. All security modules share the standard API used by the message processing subsystem.
- **Message processing modules:** it handles message-level protocol operations for current and possibly future versions of the SNMP protocol. Most importantly, they include message parsing/building and possibly invoke security services when required.
- **MIB modules:** used by the SNMP engine to maintain its configuration and operational statistics.

Data collection layer-collector agent

The collector agent has only one task: to communicate with an SNMP agent. These agents receive all the information needed from the preceding layers: the SNMP community name and the node target. The work of these agents is entirely reactive, as they only respond to the demands of the second layer. They manage errors that can be generated during SNMP communication and report them to the next layer; the algorithm overview is shown in Algorithm 1. In order to get the expected values, some parameters are needed to identify certain core switches in the network. Some of the given parameters are switch number, rack number, telecommunications enclosure room ID and its respective floor number, and building number of the university campus to which the device belongs (according to the building nomenclature). All these fields have three descriptive characters.

As shown in Algorithm 1, the work of each collector agent requires parameters such as lan, network scope, and SNMP community. The algorithm starts with a loop for every vlan on the switch. Each vlan is isolated from the others and, to make a more realistic measurement, all available vlans must be analyzed. Each of these changes the SNMP community, as a complement to SNMP validation. As a result, this validation is as follows: SNMPCommunity@VLAN_NUMBER. Communication problems may occur. Each SNMP request that files are logged along with the switch IP address and the associated vlan. Once the work is finished, the agent moves on, and the result is taken by the consolidator agent.

Algorithm 1 Collector Agent

Require: IP Address - SNMP community - Vlans - Log File
  Move
  for Every Vlan **do**
    Create connexion SNMP
    Manipulate SNMP Response
    Report missed calls
  end for
  return Number of users

Listing 1: SNMP request

```
error_indication, error_status, error_index, var_bind_table =
    ↪ cmd_gen.nextCmd(cmdgen.CommunityData('{}@{}'.format(
    ↪ community_name, str(vlan))),  cmdgen.UdpTransportTarget
    ↪ ( (target, 161), timeout=1,retries=1 ),'
    ↪ 1.3.6.1.2.1.17.4.3.1.2')
```

The nextCmd method provided by the Pysnmp library allows to perform a snmpwalk operation by iterating over the MIB with *OID 1.3.6.1.2.1.17.4.3.1.2*. The "*dot1dTpFdbPort*" allows reading a table with the MAC addresses that have been seen through a given port. As we make use of an SNMP walk operation, we do not need to specify a port, but we go through all possible ports to get an accurate measurement.

Consolidation layer-consolidator agent

Agents in this layer are tasked with getting all the network devices to be scanned and deploying collector agents. This implementation, as shown in Algorithm 2, considers a pool of threads, each collector agent will manage its own thread, and the consolidator agent is responsible for synchronizing and managing the threads. The consolidator agent will call each agent, assign them a target and manage their response. The target is an IP address of the network devices. Each one of these threads has an independent execution. Communication between these execution time agents is asynchronous; hence, we can make use of distributed parallel processing at jobs (mobile agents), thus reducing network delay and execution time. When the collector agent completes its work, the consolidator agent has to examine the data. In this step, the consolidator agent logs some information in terms of performance or errors, with the objective of making statistics. Therefore, the data is sent to the presentation layer. The logic of this agent is presented as follows.

This agent receives information from the presentation layer as the SNMP community and the name of the file source. These parameters are given to the collector agents to accomplish their tasks. Here the input file is analyzed with a mapping between physical buildings, IP references and vlans. For each physical building, a log file is created with a pool of threads. Each thread is going to point to a network device in that building. After the execution of the mobile agents' collection is done, the results are logged and added to the statistics. The final data is exposed to the presentation layer. The input file is represented as follows:

Algorithm 2 Consolidator Agent

Require: List IP Addresses - List Vlans - SNMP Community
  for IP address in list **do**
    Create thread
    Call Agent Collector
    Collect result
  end for
  Sum up results
  Get statistics
  return

- **Physical section**: Section code and name, switches in this section together with uplink port
- **Mapping**: Buildings names and codes, network device count, number of buildings
- **List of vlans**: Name, range IP.

Listing 2: SNMP request

```
1 for building in data:
2   with open(file_name, "a") as log_file:
3         log_file.write("Building: {} \n".format(building))
4   result = []
5   args = [[self.community_name, x, list_vlans, file_name,
        ↪ faults]
6   for x in data[building]]
7   with mp.Pool() as pool:
8           for i in pool.imap_unordered(Collector.get_users,
                ↪ args):
9                 result.append(i)
```

In this part of the code, we iterate over the list of buildings, logging the obtained data. In line 7, a pool of collector agents are called. This call acts asynchronously using all the processors available in the machine through the PySNMP library. In line 8, the call to the collector mobile agents is made along with the information related to the device managed by SNMP.

Application layer-application agent

This layer is responsible for initializing the system and displaying the results, as shown in Algorithm 3. During the initialization, the consolidation agents are called and assigned their tasks. Once all the underlying layers (collection and consolidation) have finished their work, they send the results to the presentation layer, where they are computed, and a HTML file is created. This file contains the HTML code necessary to display a heat map, helped by the *OpenLayers* open source technology. To complete the map, the application agent takes the coordinates and publishes the information as a web server. Thus, all the University's buildings can be appropriately named and pointed in this map. These coordinates were taken in the geodetic coordinate system 'EPSG:4326' in order to make the data available for future use. Styles and actions modules (javascript) were created with a user-centered design approach. The web module is done, and a python script is responsible of serving web files to make them available over the network. The result is shown in Figure 6. The gradient color has a variation from 0 to 1 according to the *OpenLayers* specification. The weight of each point on the map is calculated between the number of users present in the whole building versus the total number of people in the university, with red being the closest to 1, turning greener the farther away they are.

Algorithm 3 Presentation Agent

Require: Coordinates file, SNMP Community, Input File
  Get Input parameters
  Initialize MAS system
  Initialize consolidator agent
  Get physic coordinates
  Get consolidation layer data
  Create javascript heat map
  Create HTML file

Use case

Monitoring a complex system such as a campus network requires mechanisms that efficiently verify the status of users and services on the network. For this experiment, the capacity of the proposed mechanism will be measured by verifying the number of connected users on campus. The empirical experiment was carried out in the facilities of Universidad Nacional de Colombia (Bogotá), which currently has 152 buildings, out of which 129 belong to the faculties, and the remaining 23 correspond to cultural and extension centers located at different points of the city. These facilities have an average of more than 40 000 end users, including undergraduate, postgraduate and extension students, professors, administrative staff, and the general public. At the data network level, there are 73 buildings interconnected to the wired network (optical fiber and ethernet), which have approximately 30 000 structured cabling points, and are distributed in 210 telecommunications enclosures. There are 620 network core devices (switches and routers), which provide approximately 15 000 wired network connections. The university has network management and monitoring systems, such as HP Network Node Manager and Cisco Prime, which are used to monitor network problems such as equipment failures, network loops, or resource saturation (memory, processor, or channel bandwidth) on network equipment. The information provided by these systems is consolidated, which includes the total number of connected users or the distribution of users per vlan. However, obtaining particular information as to how many users are connected in a building or on a specific floor is not possible in real time, so other systems are required to analyze the data, and more processing time is spent. With SNMP, it is possible to consult the network from the core equipment network by analyzing the information of the connected users with an SNMP walk agent. However, the time consumption and processing are too high for the network's monitoring needs.

To monitor and manage the quantity of users in a specific building or floor, the consolidator agent sends the collector agents from the server located in the main data center and checks the network management devices in each building. The second phase is to analyze the information form the collector agents and create the tables and data useful to

publish them on the web server, which managed by the application agent.

As shown in Figure 3, the experiment was deployed on Vmware vsphere 6.5, on a virtual server with 4 vcores (Physical core Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz), 16 GB RAM, and 100 GB of storage. The operative system was Centos 7. The server works as a web server managed by the application agent; the MAS deployed on python uses the specific agent roles to first send the collector agents as a thread on the network. In some cases, the agent can be deployed and cloned on the server or embedded devices near the net device to collect the information. Mobility is limited, and it is possible to an ambient as a destination with an exposed port (the collector agent needs to know the route). Next, the collector agents return to the server or send the information collected from the net core device, *i.e.* the query on the MIB. The information is analyzed by the consolidator agent, and it creates information files to present on the application layer as a heat map and information table on the web server.

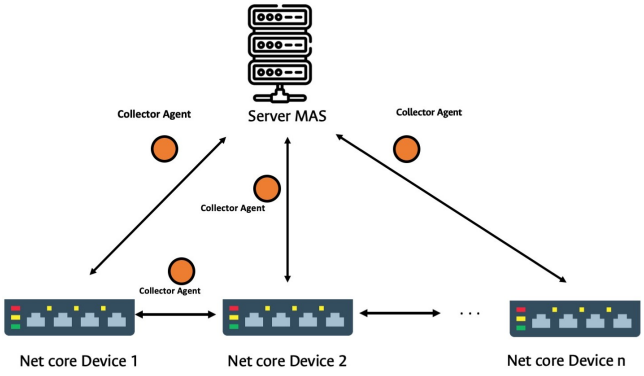

Figure 3. Infrastructure model.
Source: Authors

Results

The results of this empirical approach presented below are not comparable with other schemes proposed in the literature, such as those presented by Isaza *et al.* (2007), which are based on simulation environments. In the case of this work, it is an implementation that raises the technical rigor of the unloading of this system as a service without depending on frameworks such as JADE or Pangea, which are limited by the simulation scenario, that is, abstraction at the software level. In our case, the multi-agent system prototype used for network monitoring was implemented and deployed in computer equipment and network devices such as switches or a router.

The data to carry out the experiment was taken from two sources. The first is the legacy system that uses SNMP2 and sequentially goes one device at a time collecting data. The second source is the proposed multi-agent system model, where data are collected in parallel on the device. Measurements were made every hour, running both systems and measuring execution times. The results obtained from multiple executions for three months of data are represented in graphs to facilitate comprehension.

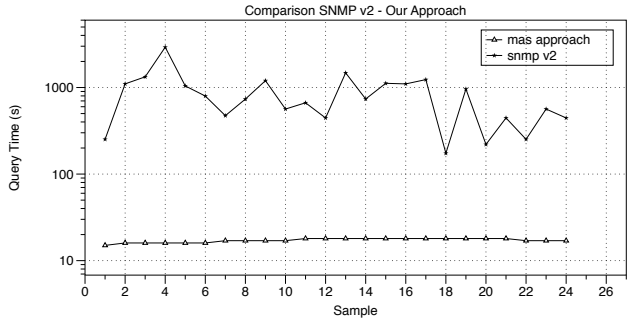

Figure 4. Time reduction; our approach (software agent approach) vs. the traditional mode (SNMP v2).
Source: Authors

In Figure 4, the analyzed data show how time is reduced because of the proposed architecture: a massive amount of agents going through the network with separated zones. Due to the way in which SNMP acts within the network, the time could be extended due to network traffic. This network traffic may cause some SNMP errors that affect the measurements and a certain number of users. Node query time with the traditional SNMP process walk takes more than one hour. In contrast, our approach with a multi-agent system architecture with some mobility reaches averages of 21,83 minutes on the wired campus network (122 network core devices at 73 buildings). During that period, the consolidation process, map creation, and publishing take place.

High traffic on the network and restrictions in the communications channel are some of the factors that make the collection time variable. In this sense, measurements take longer when there are more requests for navigation and service access by users, thus generating an additional load for the agents deployed over all the network devices registered in the system. This non-controllable variable is characterized in Figure 4, where we have the average measurements of the working days of the week. This behavior is close to the self-similar traffic. This sample allows additional management decisions to be made during this period of the day.

Other useful information to characterize user behavior is taken from the collector agents, such as the normal state of the wired network on campus. This information allows determining the amount of users per hour, as show in Figure 5, with a peak of 6 500 users linked to the network at noon. It was taken on work days in a regular academic semester for three months. This information allows planning of network resources, like switches, vlans, or management systems, infrastructure changes, and new service design. It can also be used to identify anomalous behavior as a security thread.

All types of agents in the model are designed to be customizable, this gives them the ability to make more queries on other specific MIB. This feature allows scaling the solution and taking even more data from the equipment registered in the system, not only the number of users, as seen in Figure 5. The information collected by the system peaks at noon.

This behavior allows characterizing the network and serves as input for a fault detection system or to detect anomalies that can be categorized as cyber-attacks. Finally, the information is displayed on the web server and published on the University's LAN, as shown in Figure 6.

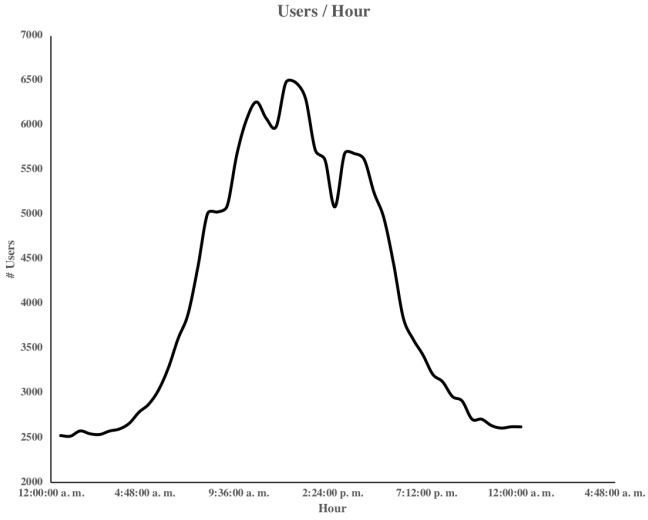

Figure 5. Mean Users campus linked to the network (data from three months).
Source: Authors

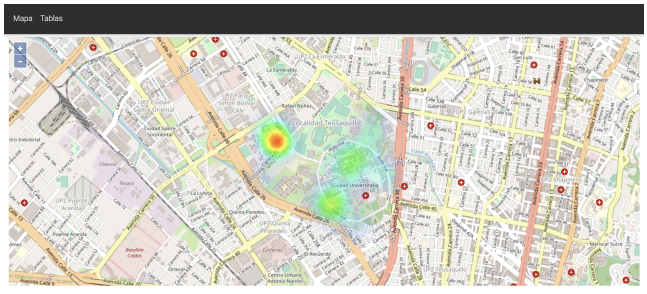

Figure 6. Application Agent (left: heat map; right: user detail table).
Source: Authors

Conclusions

In this work, we used a multi-agent system with some mobility characteristics to determine the number of users on the campus network and the density in specific buildings, reducing the data collection time by one third of the normal time. The agents were used to collect, analyze, and present information about the campus network in less time than traditional SNMP queries.

The obtained results also allow describing user behavior on campus. This information is useful as an input for modelling normal traffic for planning and decision-making for scaling and management of the campus network.

All types of agents in the model are designed to be customizable, this gives them the possibility of making more queries about other specific MIBs. This feature allows scaling the solution and taking even more data from the equipment registered in the system, not just the number of users. The model would even allow the simultaneous use of several types of collecting agents, each one collecting data for a specific problem to be solved.

We demonstrated the effectiveness of our solution on a case study by implementing the MAS system on a campus network. Our approach allows using other tools apart from the the SNMP protocol for the monitoring or analysis of telecommunications networks. This design based on multiple agents is scalable to allow the incorporation of other protocols in the collection and analysis of the data to convert it into information or present it in the form of a report or graphical interface. It can be used at the level of energy consumption calculation, in a similar way as those presented in the literature (Laštovička *et al.*, 2020; Jin *et al.*, 2019; Allahham and Rahman, 2018), with the great difference of reducing the number of sensors required, that is, the hardware component is reduced through the estimation of energy levels based on theoretical models, and some agent of the developed system is in charge of computing them.

Traditional network management and monitoring schemes made up of the management server and manageable devices, based on the SNMP protocol, have allowed different manufacturers to guarantee a monitoring scheme, offer specialized products, and create their own MIBs. However, as the number of links and core devices within the network grows, latency becomes an additional element that constitutes a limitation for managing these resources, determining the status of the network, and having a complete view of a system. For this reason, the scheme, combined with multiple agents and parallel computing, allows making a network management scheme based on distributed architectures, applications, and algorithms that allow optimizing the use of network protocols, programming language libraries, and operating systems.

This approach increases the impact and reach of these processes, and it is the novelty of this prototype. In the same way, it is not limited or non-scalable like proprietary monitoring platforms such as Nagios or Cisco Monitoring, which only use the SNMP protocol to review records and some states of network devices.

Finally, thanks to the customization capacity of the agents, it is possible to define write-type agents, which would allow the system to modify parameters in the network to improve performance or prevent issues. As for future improvements to the model, focusing on the mobility of the consolidator agent is proposed, thus allowing the coexistence of several of these agents, allowing them in turn to run on different devices in the network. This would facilitate the use of the new capabilities of network equipment that allow running Python on their operating systems, which would most likely further reduce data collection times.

References

Affandi, A., Riyanto, D., Pratomo, I., and Kusrahardjo, G. (2015). Design and implementation fast response system monitoring server using simple network management protocol

(snmp). In IEEE (Eds.) *2015 International Seminar on Intelligent Technology and Its Applications* (*ISITIA*) (pp. 385-390). IEEE. 10.1109/ISITIA.2015.7220011

Al-Naymat, G., Hambouz, A., and Al-Kasassbeh, M. (2019). Evaluating the impact of feature selection methods on snmp-mib interface parameters to accurately detect network anomalies. In IEEE (Eds.) *2019 IEEE International Symposium on Signal Processing and Information Technology* (*ISSPIT*) (pp. 1-6). IEEE. 10.1109/ISSPIT47144.2019.9001882

Allahham, A. A. and Rahman, M. A. (2018). A smart monitoring system for campus using zigbee wireless sensor networks. *International Journal of Software Engineering and Computer Systems* (*IJSECS*), *4*(1), 1-14. 10.1109/ISSPIT47144.2019.9001882

Awang, N., Ganthan, A., Samy, L. N., and Hassan, N. H. (2020). A review on risk assessment using risk prediction technique in campus network. I*nternational Journal of Advanced Trends in Computer Science and Engineering, 9*(13), 251-257. 10.30534/ijatcse/2020/3891.32020

Barriuso, A. L., Villarrubia-González, G., de Paz, J. F., Lozano, A., and Bajo, J. (2018). Combination of multi-agent systems and wireless sensor networks for the monitoring of cattle, *Sensors, 18*(1), 108. 10.3390/s18010108

Barth, W. (2008). *Nagios: System and network monitoring*. No Starch Press.

Brattstrom, M. and Morreale, P. (2017). Scalable agentless cloud network monitoring. In IEEE (Eds.) *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing* (*CSCloud*) (pp. 171-176). IEEE. 10.30534/ijatcse/2020/3891.32020

Chiandone, M., Dalle Feste, M., Bosich, D. and Sulligoi, G. (2019). Real-time monitoring and control system for trieste university campus electrical distribution grid. In IEEE (Eds.) *2019 IEEE Milan PowerTech* (pp. 1-5). IEEE. 10.30534/ijatcse/2020/3891.32020

Franco, O. H., Castillo, L. F., Corchado, J. M., and Lopez, C. A. (2007). Multiagent system for software monitoring and users' activities in a network equipment. *Scientia et Technica, 1*(34), 387-393. 10.3390/s140610804

Free SNMP Walk Tool for Windows and Linux (n.d.). https://www.manageengine.com/products/mibbrowser-free-tool/

Garcia, F. P., Andrade, R., Oliveira, C. T., and de Souza, J. N. (2014). Epmost: An energy-efficient passive monitoring system for wireless sensor networks. *Sensors, 14*(6), 10804-10828. 10.3390/s140610804

Gonçalves, P., Oliveira, J. L. and Aguiar, R. (2012). A study of encoding overhead in network management protocols. *International Journal of Network Management, 22*(6), 435-450. 10.1002/nem.1801

Hidalgo, F. and Gamess, E. (2014). Integrating android devices into network management systems based on snmp. *International Journal of Advanced Computer Science and Applications, 5*(5), 1-8. 10.14569/IJACSA.2014.050501

Hwoij, A., Al-kasassbeh, M., and Al-Fayoumi, M. (2020). *Detecting network anomalies using rule-based machine learning within snmp-mib dataset*. *arXiv* preprint. https://arxiv.org/abs/2002.02368

Isaza, G., Mejía, M. H., Castillo, L. F., Morales, A., and Duque, N. (2012). Network management using multi-agents system. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal, 1*(3), 49-54. 10.14201/ADCAIJ20121314954

Jin, Y., Tomoishi, M., and Yamai, N. (2019). Anomaly detection by monitoring unintended dns traffic on wireless network. In IEEE (Eds.) *2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing* (*PACRIM*) (pp. 1-6). IEEE. 10.1109/PACRIM47961.2019.8985052

Kaushik, A. (2010). Use of open source technologies for enterprise server monitoring using snmp. *International Journal on Computer Science and Engineering, 2*(7), 2246-2252.

Laštovička, M., Husák, M., and Sadlek, L. (*2020*). Network monitoring and enumerating vulnerabilities in large heterogeneous networks. In IEEE (Eds.) *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium* (pp. 1-6). IEEE. 10.1109/NOMS47738.2020.9110394

Li, X. and Jiang, T. (2014). Design and implementation of the campus network monitoring system. In IEEE (Eds.) *2014 IEEE Workshop on Electronics, Computer and Applications* (pp. 117- 119). IEEE. 10.1109/IWECA.2014.6845571

Madi, N. and Alkasassbeh, M. (2019). *Collecting mib data from network managed by snmp using multi mobile agents*. *arXiv* preprint. https://arxiv.org/abs/1909.02547

Min, W. (2011). Distributed network resources monitoring based on multi-agent and matrix grammar. In IEEE (Eds.) *2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming* (*PAAP*) (pp. 136-140). IEEE. 10.1109/PAAP.2011.25

Moreira, R. S., Morla, R. S., Moreira, L. P., and Soares, C. (2016). A behavioral reflective architecture for managing the integration of personal ubicomp systems: automatic snmp-based discovery and management of behavior context in smart-spaces. *Personal and Ubiquitous Computing, 20*(2), 229-243. 10.1109/PAAP.2011.25

Moura, P., Moreno, J. I., López-López, G., and Álvarez-Campana, M. (2021). Iot platform for energy sustainability in university campuses, *Sensors, 21*(2), 357.

Pashamokhtari, A., Gharakheili, H. H., and Sivaraman, V. (2020). Progressive monitoring of iot networks using sdn and cost-effective traffic signatures. In IEEE (Eds.) *2020 Workshop on Emerging Technologies for Security in IoT* (*ETSecIoT*) (pp. 1-6). IEEE. 10.1109/ETSecIoT50046.2020.00005

Pysnmp 4.4.12 (n.d.). *Pysnmp 4.4.12*. https://pypi.org/project/pysnmp/

Sánchez, R., Herrero, A., and Corchado, E. (2013). Visualization and clustering for snmp intrusion detection. *Cybernetics and Systems, 44*(6-7), 505-532. 10.1080/01969722.2013.803903

Stallings, W. (1998). *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison-Wesley Longman Publishing Co., Inc. 10.1109/COMST.1998.5340405

Torre, G. D. L. and Yucelen, T. (2018). Adaptive architectures for resilient control of networked multiagent systems in the presence of misbehaving agents. *International Journal of Control, 91*(3), 495-507. 10.1080/00207179.2017.1286040

Wan, S., Lu, J., Fan, P., and Letaief, K. B. (2017). To smart city: Public safety network design for emergency. *IEEE Access, 6,* 1451-1460. 10.1109/ACCESS.2017.2779137

Yang, C., Liu, J., Kristiani, E., Liu, M., You, I., and Pau, G. (2020). Netflow monitoring and cyberattack detection using deep learning with ceph. *IEEE Access, 8*, 7842-7850. 10.1109/ACCESS.2019.2963716