SNMP Agent for SR Linux

Guilherme André Soares Cardoso
Nº 98495
guilherme.soares.cardoso@tecnico.ulisboa.pt

Instituto Superior Técnico

October 2022

Abstract In recent years computer networks have evolved by leaps and bounds, progressing based on factors such as increasing complexity, size, security, virtualization, and use in data centers. Due to this sudden expansion, technologies developed for managing and monitoring network devices have become increasingly unadapted to the current context of computer networks. Faced with this mismatch, the industry is looking for alternatives that allow automation and functional monitoring of networks. Nokia's SR Linux operating system is an open system dedicated to data centers networks that use modern management interfaces such as gNMI or JSON-RPC.

This master's dissertation describes the development of an SNMP agent for the SR Linux system. The agent aims to send SNMP Traps when an element of the system that is being monitored change state according to certain conditions. This agent uses recent technologies such as gNMI, for the internal subscription of the elements to be monitored, and Yang for easy integration into the SR Linux system. The implementation of the agent was written in python and specific libraries were used, as is the case of *pygnmi*. The implementation of the agent was tested in a virtual environment, using different scenarios, which allowed us to evaluate its correct operation.
This dissertation was supported by Nokia and Instituto de Telecomunicações.

Keywords: Network Management · Network Telemetry · SNMP · YANG · gNMI/gRPC · JSON-RPC

1 Introduction

The Service Router Linux system, better known as SR Linux, is the latest solution developed by the telecommunications company Nokia for routers and switches in data centers. This system is based on Linux, which improves reliability, portability, and the creation of Linux-based applications. The applications are called agents in SR Linux.

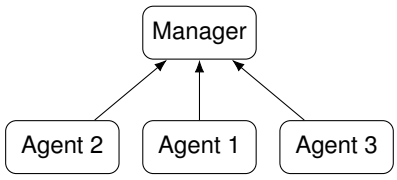
This new operating system was designed to be part of Nokia's *Datacenter Fabric Solution*. This solution, which is based on the use of the SR Linux system, aims to present a scalable solution, an increase in automation between the different components, a DevOps mechanism, simplifying the development and integration of applications in the various components of the solution, management of work between components simply and effectively, as well as improving monitoring of each component of the solution.

The SR Linux system is based on the model-driven CLI management model, that is, CLI commands follow and operate according to data models written in YANG [1]. The use of YANG data models, in the way of observing the system, increases the consistency between the different technologies of management and monitoring of the networks, namely, gNMI/gRPC and JSON-RPC. These technologies allow the management and collection of data efficiently using new mechanisms at the transport and content layer, compared to older technologies such as SNMP.

2 Theoretical Background

2.1 SNMP

The SNMP protocol is an application layer protocol, developed by IAB and defined in RFC 1157 [2]. This protocol was developed in the late 80s and allows the management and configuration of network devices from an NMS, which we will call manager.



Figures 1: Architecture manager-agent

The protocol uses a manager-agent [3] architecture, where the various agents communicate with the manager. The SNMP protocol is based on three basic components [4]:

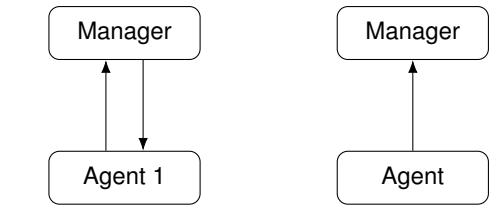
- Agents — Consists of software that is installed on the network devices to be managed. The main responsibility of the agent is to communicate the information contained in the MIB to the manager when requested. It can, when properly configured, send messages like SNMP Trap when some MIB element changes state.

- Managers — Send messages to agents to collect information about their MIBs.

- MIB — It is a database composed of two object formats: scalar or tables, which exist in each agent and which represent the device's data models.

This protocol uses the notation SMI [5] to define its MIB [6]. Data representation exists in two ways: scalar or by tables. The scalar form represents an instance of an object, while the table form allows representing and correlate multiple instances of objects, which are grouped in MIB tables. Both forms are identified by OID. These OID have a structure that represents a path along a normalized data tree called ISO Object Identifier Tree.

The SNMP protocol has two modes of action, as illustrated in Figure 2.



Figures 2: SNMP protocol operation modes

In Request/Response mode, the manager initiates communication with the agent, sending a request to the agent. On the other hand, in Trap mode, the agent is the one that initiates the communication with the manager.

2.2 NETCONF

The NETCONF is a network device configuration protocol, published by the IETF Working Group at the end of 2006 and defined in RFC 4741 [7]. This protocol presents a set of operations that allow obtaining and configuring data on network devices through Remote Procedure Calls. The protocol uses a client/server architecture, where the server is the network device and the client is the system that operates the network device through the operations provided by NETCONF.

2.2.1 Datastores

The NETCONF protocol introduces the concept of datastores. These datastores are sets of settings that allow a device to change state. Three types of configurations have been defined, and more can be added:

- running — Active configuration on the device;
- startup – Settings that are loaded by default when starting the device;
- candidate – Configuration that allows you to change the configuration of a device without changing the current state of the device.

2.3 gNMI

The gNMI framework is a specification of the gRPC protocol developed by Google. This framework aims to offer network administrators a tool capable of configuring and monitoring network devices remotely [8].

This framework uses Protocol Buffers to define the service it provides, as well as the messages that must be exchanged. The specification defines four methods: Get, Set, Capabilities and Subscribe. These methods are used on the client to invoke functions with certain parameters and return type predefined in the service. The methods that are defined in the service have the following functionalities:

- Capabilities – Allows the client to understand what data models the server supports, as well as encodings and the gNMI version the server uses.

- Get – Gets the state/value of a given system element identified by a path according to the gNMI Path Convention.

- Set – Modifies the state/value of a given system element identified by a path according to the gNMI Path Convention. This function allows three different operating modes: Update, Replace and Delete.

- Subscribe – It allows subscribing to a system element so that when it changes its status/value, the customer receives a notification about the status change. This function allows three different operating modes: STREAM, POLL and ONCE.

The Subscribe method implements the bidirectional streaming mechanism. This method is used for subscription-based network device monitoring (streaming telemetry), where NMS sends a request to the network device to subscribe to state changes of a certain path. The method provides three different operating modes:

- ONCE – In this operating mode, the gNMI client sends a request to the gNMI server, which responds with the desired data and then closes the communication channel between the two. It is an operating mode similar to the Get function, but it presents better performance when you try to get a large object from the server.

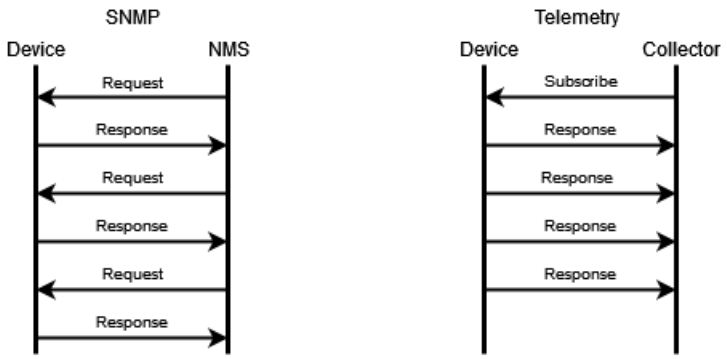
– POLL – In this operating mode, the gNMI client sends requests to the server in streaming. That is, the client wants to acquire information about a certain path, sends these requests, and then the server responds with the desired information. Unlike the ONCE operating mode, the channel remains open, allowing the customer to place more orders.
– STREAM – In this operating mode, the gNMI client sends a request to the gNMI server with the subscriptions it wants to configure and/or monitor (i.e., which paths it wants to receive information). This information can be obtained based on events (ON_CHANGE) or on time intervals (SAMPLE). Unlike the POLL operating mode, the client only sends a subscription request at the beginning of the TCP connection, not being able to send more requests in that TCP session.

2.4 YANG

YANG is a data modeling language was originally developed to model the content layer of the NETCONF [7] protocol. The content to be modeled and described in version 1.0 of the RFC 6020 language [9] was the configuration data and state data of a network device, the remote procedure calls that could be invoked and the notifications.

2.5 Network Telemetry

With the evolution of networks, network monitoring technologies such as SNMP, Syslog or CLI have become increasingly out of step with current reality. With this premise, the concept of Network Telemetry or also known as Streaming Telemetry was developed. In this concept, described in RFC 9232 [10], telemetry data are obtained based on subscriptions of certain elements of a network device, instead of queries/polling, as in SNMP.



Figures 3: SNMP operating mode and Network Telemetry

In Figure 3, we can see the different concepts. On the left, the common SNMP operating mode. On the right, the operating mode of this new concept.

3 Proposed Architecture

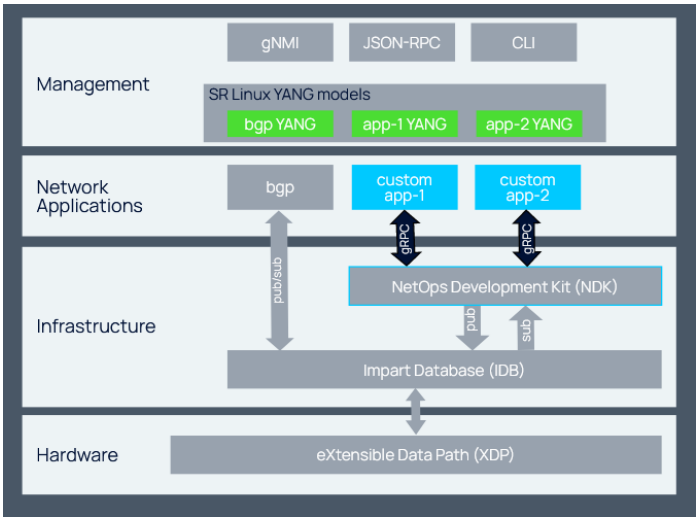
3.1 Requirements

The main objective of this dissertation was to develop an SNMP agent. In developing this agent, the following criteria were taken into account:

- The SNMP agent will only have the operational function of sending SNMP Traps, and it will not respond to any other operation provided by the SNMP protocol.
- The agent must be configurable from the gNMI, JSON-RPC, and CLI management interfaces. For this, it will need a data model in YANG that allows the correct management of the agent.
- The agent will have to present a structure capable of representing an event. That is, a structure that explains under what conditions an SNMP Trap is sent, as well as other information useful to the event.
- The agent will have to support sending traps to multiple destinations.

3.2 Architecture of the SR Linux system

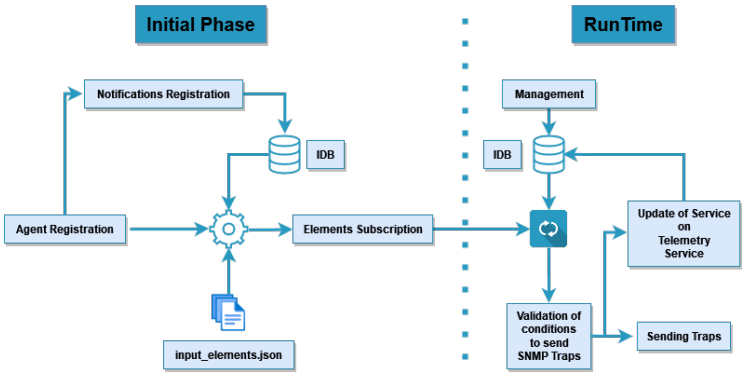
The SR Linux operating system provides an application development engine (SDK) called NDK. From this NDK, it is possible to integrate agents in the SR Linux operating system with other native or non-native applications. The interaction between the agent and the rest of the system is possible through the NDK gRPC service that exposes a set of RPC methods that allow interaction with the rest of the system, even at a low level, through the IDB. The IDB is responsible for the communication of information between the different agents and the system, and stores the information in protobufs. We can observe these interactions in Figure 4 [11].



Figures 4: Integration of agents in the SR Linux system

3.3 Architecture of the proposed solution

In Figure 5, we can see the execution flowchart of the idealized SNMP agent, the components will not be detailed in this document.



Figures 5: Architecture of the agent

This solution is divided into two execution phases: i) the Initial Phase, where the initial configurations are introduced in the agent; ii) the RunTime Phase, which consists of an infinite loop (represented in the Figure by the blue symbol on the right), while the agent is running.

In the proposed solution, the concept of an element was introduced. That is, each SNMP Trap is associated with an element that is intended to be monitored. This element is represented by a JSON object that has different parameters that allow the functional mapping of the gNMI subscription to SNMP Trap.

In the Table 1, we can see the structures of an Element and a Target in a tree diagram.

```
snmp-agent
+-- monitoring_elements
|  +-- element [resource]
|     +-- parameter
|     +-- monitoring_condition
|     +-- trigger_condition
|     +-- trigger_message
|     +-- resolution_condition
|     +-- resolution_message
|     +-- trap_oid
|     +-- resource_filter
```

```
snmp-agent
+-- targets
   +-- target [address]
      +-- network-instance
      +-- community-string
```

Table 1: Structures of an Element and a Target

4 Evaluation

The evaluation of the agent is not based on quantitative methods (e.g., measurement of the SNMP Trap sending time), but on the verification of the correct functioning of the features that the agent has. Taking into account the proposed requirements for the development of the agent, the evaluation is mainly related to the verification of the following functionalities:

- Mapping of the gNMI subscription and the event that occurred;
- Sending SNMP Traps;
- Consistency between the configuration file, the running datastore and state;

4.1 Scenarios and Conditions

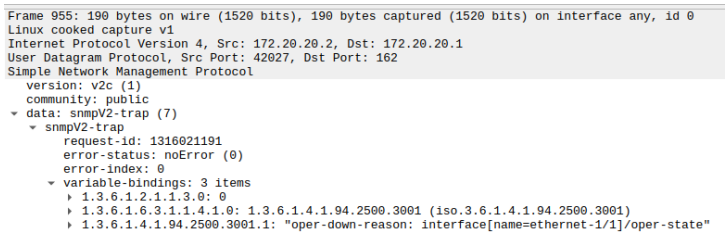
To verify the correct functioning of the agent, three different simulation scenarios were proposed:

- **Scenario 1** - An SR Linux device connected to a Collector. In this case the Collector will be the host that runs ContainerLab.
- **Scenario 2** - An SR Linux device, running the agent, connected to different SR Linux devices, where the first device (the one that runs the agent) is connected to a Collector. In this case the Collector will be the host that runs the ContainerLab.
- **Scenario 3** - An SR Linux device, running the agent, connected to three Collectors. One of them is the host that runs ContainerLab.

4.1.1 Results

For reasons of document simplification, only the results of scenario 1 will be shown. The results of the other scenarios are documented in detail in the dissertation.

In this scenario, the Collector is connected to the SR Linux through the management instance network (mgmt). Collector is the host that runs ContainerLab and runs the *snmptrapd* tool to listen for the SNMP Traps destined for it. Thus, to verify the correct functioning of the agent for this scenario, the agent was configured with an element that allows sending an SNMP Trap to the Collector when an interface goes down(i.e., the oper-state of the interface goes from up to down). Under these conditions, the simulation was performed by turning off the ethernet-1/1 interface of SR Linux, thus obtaining the appropriate simulation scenario to trigger the event.



Figures 6: Simulation Result for the Scenario 1

Figure 6 was captured in Collector and shows that the SNMP Trap was correctly received. Through this Figure, we can observe the correct use of the destination IP address (Collector address — 172.20.20.1) and the correct use of the destination port (162). In the SNMP layer of the packet, we can observe the use of the community string, the version used, as well as the variable bindings. In this case, this SNMP Trap presents three different variable bindings. The first variable binding concerns the timeticks and sending them is mandatory, due to the lack of this information in the agent, the value 0 is always sent in this field. The second variable binding is the OID configured in the event. The third variable binding is a string. As the .1 was added to the original OID, we know that it was triggered by the trigger_condition of the configured event, which coincides with the proposed simulation.

5 Conclusion

The main objective of this dissertation was the development of an SNMP agent for the SR Linux operating system. This agent serves as an interface between two monitoring technologies: gNMI and SNMP.

With this in mind, firstly, a research was carried out on the state of the art in the area of network monitoring. A vast area full of concepts and protocols, developed over the years in order to meet the needs of each era in which they were developed, as is the case of SNMP, NETCONF, RESTCONF, gNMI.

In addition to the study of these protocols, the concept of Network Telemetry was also studied. This relatively recent concept in the area tries to fit these technologies and concepts into a framework for an easier interpretation of the flow and consumption of telemetry data in network topologies.

Based on the concepts and protocols studied, an agent execution flowchart was established. This execution flowchart would have to take into account the integration with the SR Linux NDK, and therefore a study on the SR Linux system itself was carried out.

In the design of the agent, we tried to make it as generic as possible, and therefore it is possible to dynamically add events to the agent. The other solution would be to fix which elements of the system the agent would be listening, and the agent would only send the SNMP Traps for those events. As for the agent's performance, it was found that it fulfilled all the functional requirements proposed for this dissertation.

In short, this dissertation allowed the study and development of an application (agent) in a structured operating system designed with the current technologies and facing the modularity of the system itself, such as the use of MD-CLI and YANG data models as a way to see the different functionalities of the system, but which needs to support older technologies, in this case SNMP, for a correct integration in existing topologies.

References

1. Karneliuk. Model-driven command line interface (md-cli) in nokia sr os 16.0, Mar 2021.
2. Jeffrey D. Case, Mark Fedor, Martin Lee Schoffstall, and James R. Davin. Simple network management protocol (snmp). STD 15, RFC Editor, May 1990. http://www.rfc-editor.org/rfc/rfc1157.txt.
3. Jürgen Schönwälde. On the future of internet management technologies. *IEEE Communications Magazine*, pages 90–97, 2003.
4. Abubucker Samsudeen Shaffi. Managing network components using snmp. *International Journal of Scientific Knowledge*, pages 11–18, 2013.
5. Keith McCloghrie, David Perkins, and Juergen Schoenwaelder. Structure of management information version 2 (smiv2). STD 58, RFC Editor, April 1999. http://www.rfc-editor.org/rfc/rfc2578.txt.
6. Marshall T. Rose and Keith McCloghrie. Structure and identification of management information for tcp/ip-based internets. STD 16, RFC Editor, May 1990. http://www.rfc-editor.org/rfc/rfc1155.txt.
7. R. Enns. Netconf configuration protocol. RFC 4741, RFC Editor, Dezembro 2006. http://www.rfc-editor.org/rfc/rfc4741.txt.
8. Openconfig. grpc network management interface. https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md, 2018.
9. M. Bjorklund. Yang - a data modeling language for the network configuration protocol (netconf). RFC 6020, RFC Editor, October 2010. http://www.rfc-editor.org/rfc/rfc6020.txt.
10. H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, and A. Wang. Network telemetry framework. RFC 9232, RFC Editor, May 2022.
11. Nokia. Ndk architecture. https://learn.srlinux.dev/ndk/guide/architecture/. Acedido a 2022-01-10.