Performance Analysis of gNMI Streaming Telemetry-Based Monitoring Systems Using Containerlab Network Simulation

Fierda Kurniacahya Ariefputra[1], Eueung Mulyana[1]

[1] Electrical Engineering Study Program, School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, Jawa Barat 40132, Indonesia

[Submitted: 24 October 2023, Revised: 27 December 2023, Accepted: 21 March 2024]
Corresponding Author: Fierda Kurniacahya Ariefputra (email: 23221081@mahasiswa.itb.ac.id)

ABSTRACT — The rapid growth of the Internet has impacted the digital service development. This surge in demand has created opportunities for digital service industry players. Despite its positive impact, the growth of the Internet also poses technical challenges. In managing the increasing data traffic, resource monitoring plays a vital role. One of the latest methods for monitoring these resources is the utilization of the Google's Remote Procedure Call (gRPC) Network Management Interface (gNMI) streaming telemetry system. While it seems superior to current protocols, there is a need for further exploration into the implementation of streaming telemetry systems. This paper specifically investigates the trade-offs and performance of gNMI streaming telemetry. The design and simulation were conducted utilizing containerlab, a Docker-based networking lab tool. In the Docker-based simulation, integration between the monitoring system and network topology was implemented. The results from observing each protocol indicate that the monitoring system's metric retrieval activity had minimal impact on network conditions. This is evident in the consistently low average network latency and nearly uniform throughput, except in instances of packet loss and congestion. Simulation observations indicate that the gNMI monitoring system utilized input/output (I/O) resources more intensively compared to other protocols. The research also examined the integration of gNMI streaming telemetry and log monitoring, revealing a 70 MB rise in memory usage and a 33% increase in Disk I/O resources. Furthermore, the study uncovered signs of a 50% increase in CPU utilization by the gNMI monitoring system compared to the average data recorded in the observations.

KEYWORDS — Network Monitoring, Streaming Telemetry, gRPC Network Management Interface, Data Centre Network, Simulation.

I. INTRODUCTION

The proliferation of the Internet usage has profoundly impacted numerous sectors, significantly enhancing people's quality of life [1], [2]. The interconnected network of data exchange has revolutionized numerous sectors, including communication, education, entertainment, and commerce. The Internet has become an integral part of everyday life, granting individuals effortless access to information and fostering global connections.

The rapid expansion of the Internet has profoundly influenced the evolution of businesses offering digital hosting and Internet services. With the continual rise in Internet accessibility, an increasing number of individuals and businesses seek dependable hosting and Internet solutions for establishing online platforms. This trend is not limited to large corporations or multinational enterprises; small and medium-sized companies operating at a national level are also seizing this opportunity, experiencing substantial growth [3]. This surge in demand has opened up significant opportunities for businesses and industries within the digital services and Internet services sectors, enabling them to expand their operations and foster growth.

Despite its many positive impacts in various fields, the rapid growth of the Internet also presents technical challenges, especially in managing communication network traffic and computing resources As the number of connected devices continues to rise, the demand for network bandwidth and data storage exponentially increases. This surge in connectivity can lead to issues like network congestion, slower data transfer speeds, and limited resources for services and applications on the hosting service [4].

One of the most modern methods for monitoring information technology infrastructure resources is by utilizing a system based on streaming telemetry. This system collects and analyzes data in real time from various network devices and computational resources. Such monitoring systems offer valuable insights into performance, utilization, and the behavior of information technology infrastructure. Streaming telemetry empowers operators to monitor network traffic patterns, identify current data constraints, and take proactive measures, particularly to optimize and ensure efficient data flow (with minimal overhead).

In traditional network monitoring approaches, resource monitoring typically relies on a Simple Network Management Protocol (SNMP) to periodically collect data [5], [6]. This method primarily focuses on monitoring network device metrics.

However, recent advancements in monitoring systems and network infrastructure, like streaming telemetry, incorporate protocols such as Google's Remote Procedure Call (gRPC) Network Management Interface (gNMI). This framework facilitates the continuous streaming of metrics, logs, and trace data from diverse sources within data centers and servers. [7], [8]. This approach can be integrated with microservice resource monitoring applications packed in virtualized environments and can be paired with continuous integration and continuous delivery/continuous deployment (CI/CD) cycles.

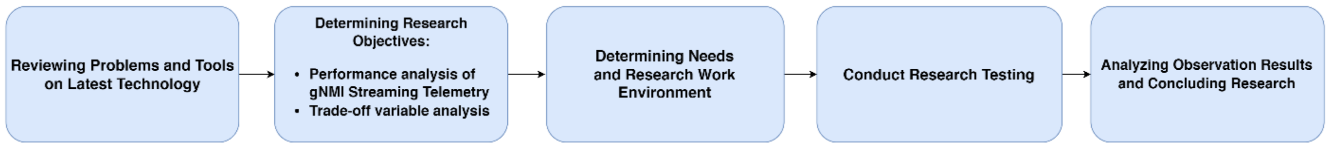
Reviewing Problems and Tools on Latest Technology
Determining Research Objectives:
• Performance analysis of gNMI Streaming Telemetry
• Trade-off variable analysis
Determining Needs and Research Work Environment
Conduct Research Testing
Analyzing Observation Results and Concluding Research

Figure 1. Research methodology.

Although it seems more sophisticated and efficient than existing protocols, further studies are needed on the implementation strategy of streaming telemetry systems. This is particularly important for understanding their performance under near real-time conditions in a distributed system environment, such as a cloud computing environment. Research on the use and implementation of each network monitoring and telemetry technology has been conducted by several experienced researchers and industry practitioners. By reviewing prior research, it is possible to ascertain the research context's focus, pinpoint gaps in earlier studies, and formulate a research design.

Several researchers have conducted research on implementing gNMI-based telemetry systems in fiber optic networks and data centers. References [9] and [10] discussed the the implementation of two technology components for transceiver management and telemetry monitoring in fiber optic networks. Telemetry monitoring was conducted using the gNMI protocol and the OpenConfig device model. The research showcases the viability of this technological approach through real-time telemetry measurements and an analysis of its scalability with varying numbers of transponder nodes. Additionally, numerous studies have explored the utilization of streaming telemetry in intricate fiber optic networks and data centers. This method aids in identifying traffic patterns, enabling the detection of state changes and soft failures within fiber optic networks [11]–[17].

In Reference [18], the performance of SNMP-based network monitoring was compared with streaming network telemetry using the gRPC framework and Google Protocol Buffers (GPB). The study results indicate that retrieving ifTable statistics encoded with Compact-GPB consumes approximately two and a half times less bandwidth compared to SNMP polling with GetBulk. Moreover, the measured round-trip network delay, encompassing packetization, serialization, and telemetry stream processing, is roughly four times lower than that of SNMP. Additionally, the measurements demonstrate that network monitoring with streaming telemetry is more CPU-efficient than SNMP polling.

Another research was undertaken to compare two network monitoring solutions: SNMP-based and Protocol Buffers and gRPC-based [19]. The findings indicated that SNMP outperformed when retrieving information about objects with a value of 1 byte. Moreover, SNMP demonstrated superior performance when retrieving metric objects with larger values, up to 26 telemetry objects per message. However, beyond this threshold, the combination of Protocol Buffers and gRPC proved more efficient, resulting in reduced data transmission (in bytes) for a given number of objects. Notably, no significant impact on router memory and CPU utilization was observed.

The research focused on the performance of streaming telemetry in monitoring network conditions and resources within a simulated data center virtualization environment. The performance in question includes the effect of the telemetry stack on host device resource utilization, the influence of the interval duration configuration, and the number of metrics subscribed by streaming telemetry. The research also studied the behavior of network traffic in relation to the monitored metric parameters. This research was conducted in a Docker-based virtualization simulation environment, with a scenario that involves monitoring resources on network devices operating in the data center using Border Gateway Protocol Ethernet Virtual Private Network (BGP EVPN VXLAN) protocol communication [20].

II. METHODOLOGY

This research aimed to fill the information gap regarding the performance information and trade-off variables of streaming telemetry as discussed in existing publications, articles, and research papers. Specifically, the trade-offs examined in this research were the duration of the scraping interval and the number of objects subscribed to by the gNMI telemetry stream. The research method is illustrated in Figure 1, which outlines the steps and workflow followed in this study. Observations were conducted by manually recording the parameter values displayed by the Glances tool throughout the observation period.

A. gRPC NETWORK MANAGEMENT INTERFACE (gNMI)

gNMI is a gRPC protocol extension developed by OpenConfig and standardized by the Internet Engineering Task Force (IETF). It provides a standard method for managing and monitoring network devices using the remote procedure call (RPC) mechanism [21]. The protocol operates by initiating a gRPC connection between the client (typically a monitoring system or application) and the gNMI server on the network device. This connection utilizes different transport protocols like Transmission Control Protocol (TCP), Transport Layer Security (TLS), or gRPC over HTTP/2. The client authenticates its identity to the server using credentials or certificates for secure communication. Upon verification of the client's identity, the server grants authorization for the requested operation based on the client's privileges [22].

The client sent gNMI requests to the server to perform various operations, including retrieving configuration data, modifying settings, subscribing to status updates, and querying operational information. These requests were specified using the gNMI Protocol Buffers API, which defined the desired action and parameters. The server received gNMI requests, validated them, and processed them accordingly. gNMI interacted with the underlying network operating system or network device drivers to execute the requested actions, such as reading or modifying configuration data. Once the request was processed, the server sent the gNMI response back to the client, containing the requested information or indicating the success/failure status of the operation. This response was encoded using Protocol Buffers.

In addition to request-response interactions, gNMI supports subscription-based mechanisms. It means clients can subscribe to receive updates or asynchronous notifications regarding

TABLE I
TOOLS AND VERSIONS IN SYSTEM DESIGN FOR RESEARCH

Tool Name	Version
Virtualbox	6.1.40 r154048
Linux Mint OS	20.2
Docker	20.10.21
Containerlab	0.42.0
Nokia SR Linux (Docker container)	23.2.2
Telegraf (Docker container)	1.27.0
influxDB (Docker container)	2.7.1
sFlow-RT (Docker container)	3.0-1676
Prometheus (Docker comtainer)	2.37.8
Grafana (Docker container)	9.5.2
gNMIc (Docker container)	2.3.0

specific data changes or events. The server then pushes these updates to the client in real-time, enabling seamless monitoring and event-driven management.

B. MONITORING SYSTEM TOOLS AND SOFTWARE SPECIFICATIONS

The system design in this study utilized several open-source software and networking tools. The use of open-source tools aimed to enhance the transparency of this research, enabling researchers and industry practitioners to replicate the scenarios presented. This also facilitated the validation and verification of the research results, ensuring that the designed system functioned as intended. Additionally, the flexibility and customizability provided by open-source tools were significant considerations in this research. These factors underpinned the decision to use containerlab for network simulation in this study.

The tools and software utilized in this research are presented in Table I, along with the respective versions employed. The hardware specifications of the computer included a 2.6 GHz Intel Core i7 CPU with 6 cores and 16 GB of random access memory (RAM). The Linux Mint virtual machine was configured with a 4-core processor and 8 GB of RAM.

C. NETWORK MONITORING STACK CONFIGURATION

This research was conducted to test the performance and utilization of monitoring systems across various protocols, focusing on their effectiveness in monitoring resource and network metrics in simulated data centers. The examined protocols were SNMP, sFlow, and gNMI.

The protocol in this section is divided into three monitoring subsystems, which consist of several tools (collectively known as the monitoring stack) to retrieve, process, and display metric data from the monitored system.

Each monitoring subsystem had a monitoring tool component with similar functions. All utilized tools ran in a separate Docker container environment, except for the sFlow-RT Docker image. The communication between tools and the data center simulation network was managed by containerlab. Each subsystem employed a different way of obtaining metrics or status from network devices.

The gNMI subsystem utilized "pull" operations to change configurations on network devices and "push" operations to send information from the server to the collecting device without a request, thereby obtaining status and metrics information from the device. The gNMI subsystem had to subscribe to a specific port on the network device to enable

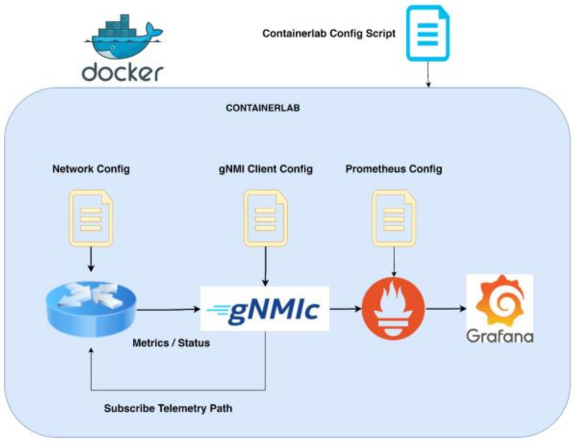

Figure 2. Architecture of gNMI subsystems in the study.

communication. This port is specified in the gNMI client configuration file, as illustrated in Figure 2.

The SNMP monitoring subsystem polled (periodically checked or sent requests to the server for updates) a specific IP address. This process aimed to communicate with network devices to request status updates and metrics of the network device. The device responded to the object identifier (OID) requested by the collector, by polling the information to the collector.

The sFlow subsystem utilized a different method compared to the other subsystems. The sFlow collector required only the Interenet Protocol (IP) address of the destination device for configuration. Subsequently, the network device transmitted all data specified in the sFlow management protocol as defined by the vendor. The employed sFlow collector was directly integrated with the Prometheus time series database (TSDB) and InMon dashboard in one Docker image. Some network configuration, Docker image configuration, and environment variables are written in the containerlab deployment configuration file as shown by the gNMI subsystem architecture in Figure 2.

D. SOCKET BUFFER CONFIGURATION

The design of the network monitoring system and network topology in this research was implemented using containerlab. Containerlab employs the Docker bridge, which was the default Docker network driver. This driver enabled inter-container communication on the same Docker host. The Docker bridge created a virtual bridge interface called "Docker0" on the host system, which acted as a virtual switch to facilitate inter-container communication. The Docker bridge used a variable memory network buffer, or socket buffer, from the host. The memory socket buffer was set with a default and maximum capacity of 32 MB, allowing it to accept jumbo-sized packets (9,000 bytes).

III. RESULTS AND DISCUSSION

The data collection for this research was conducted through direct observation of the computing resources utilized when the network transmits TCP or User Datagram Protocol (UDP) packets. The Glances tool was employed to display the resources on the host responsible for the data center network simulation and network monitoring.

A. ANALYSIS OF gNMI-BASED MONITORING SYSTEM PERFORMANCE

The findings from tests conducted using the UDP packet type are illustrated in Figure 3. The graph depicts a comparative

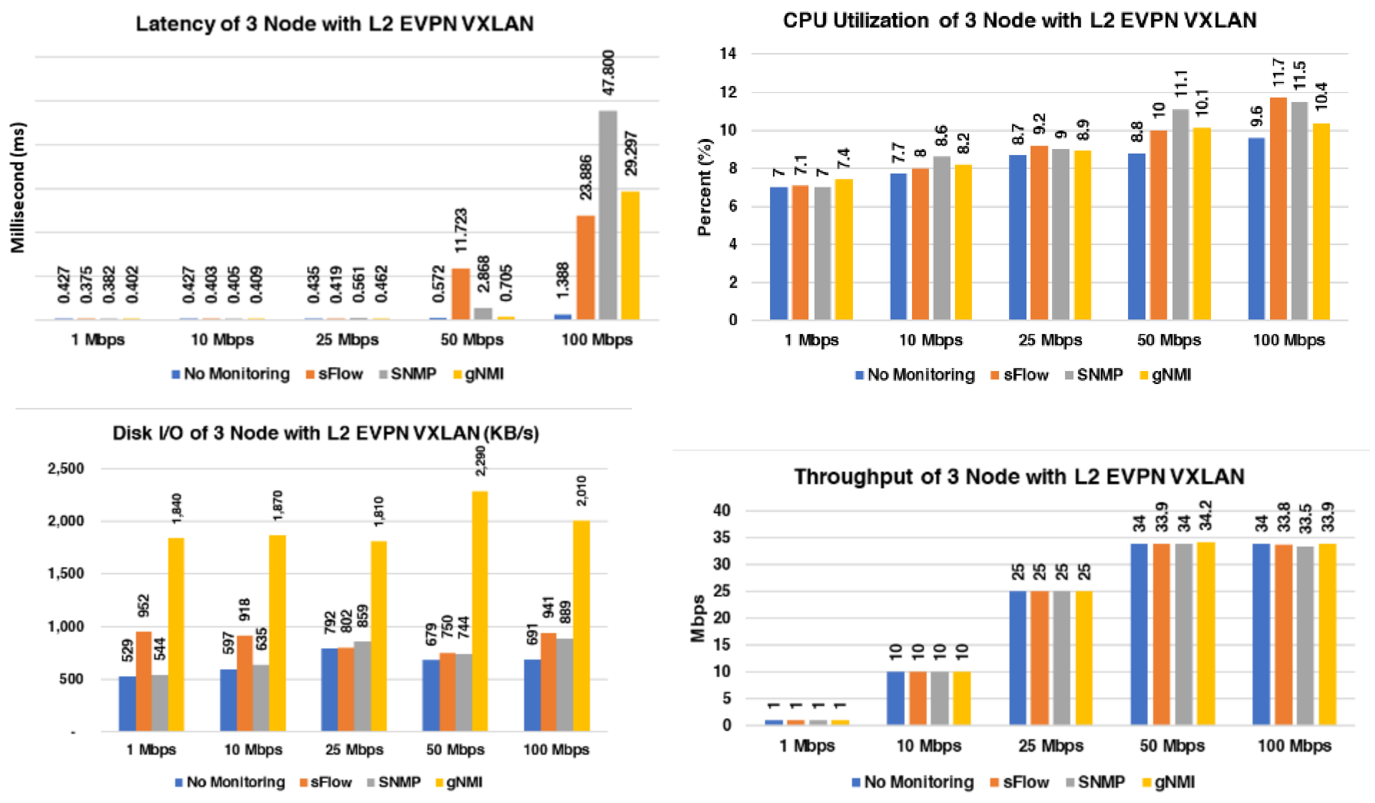

Figure 3. Test result data with UDP data traffic.

analysis of three monitoring systems utilizing metric data obtained from three Nokia Service Router Linux (SR Linux) nodes operating under the Layer 2 BGP EVPN VXLAN protocol. Owing to the constraint of the Docker-based SR Linux Network OS, capable of processing only 1,000 packets per second, assessments conducted at 50 Mbps and 100 Mbps data rates encountered packet loss, resulting in diminished throughput and elevated latency levels. This scenario continued to fulfill the objective of analyzing network conditions, transitioning from an elephant flow scenario to one characterized by lossy channels and congestion.

The observations depicted in Figure 3 regarding throughput parameters do not indicate any significant alterations in the utilization of the monitoring tool stack. However, Figure 3 illustrates that the latency parameter reveals heightened latency in SNMP and gNMI-based monitoring systems under conditions of network loss and congestion. The CPU utilization of sFlow and SNMP-based monitoring systems under such conditions demonstrated an approximate 2% increase compared to network topology without monitoring systems, whereas the CPU utilization with gNMI-based monitoring rose by no more than 1.5% under similar lossy and congested network conditions.

In the RAM usage observation (5.24 GB without a monitoring system), it was noted that the sFlow-based stack monitoring system consumed more resources compared to alternative monitoring systems, reaching up to 6.24 GB (a 1 GB increase). Conversely, both SNMP and gNMI-based monitoring systems utilized approximately 5.37 GB of RAM resources (a rise of approximately 200 MB). Regarding Disk I/O usage parameters, it is evident that the gNMI-based monitoring system exerts greater resource consumption compared to other protocols. The values depicted in the observations may exceed twice the Disk I/O of alternative monitoring systems. Subsequently, the test proceeded with traffic involving TCP packet types. The outcomes of this test are illustrated in Figure 4.

The observation of TCP packet latency in Figure 4 reveals a difference when compared to the latency of UDP packets. The average TCP latency ranged from 0.4 ms to 0.55 ms, which was not significantly different from the test results between monitoring systems and those of the topology without monitoring systems. The observations on RAM usage and CPU utilization during the testing of TCP packets did not show significant differences compared to the testing of UDP packets. Regarding the Disk I/O usage parameter, the results were consistent: the gNMI-based monitoring system utilized twice as many resources as the other monitoring systems. However, the Disk I/O resource usage value in tests with TCP-type data packets was relatively lower than in tests with UDP data packets.

Data collection in this test was conducted manually, and the recorded value was determined based on the mode value during the time intervals between the scraping activities of the monitoring system. The test was carried out with a consistent scraping interval duration of 15 s. The SNMP monitoring system was configured to retrieve five OID objects, while sFlow collected all metric objects provided by network operating system SR Linux. In contrast, the gNMI-based monitoring system was configured to retrieve nine telemetry object directories containing 65 telemetry object values. During the test, it was observed that CPU utilization could increase by 25% for the sFlow and SNMP-based monitoring systems, whereas the gNMI-based monitoring system could experience a 50% increase.

B. ANALYSIS OF DATA PULLING INTERVAL DURATION

In this test scenario, the effect of the pulling interval duration of the gNMI telemetry subscription on resource utilization and network conditions during active data

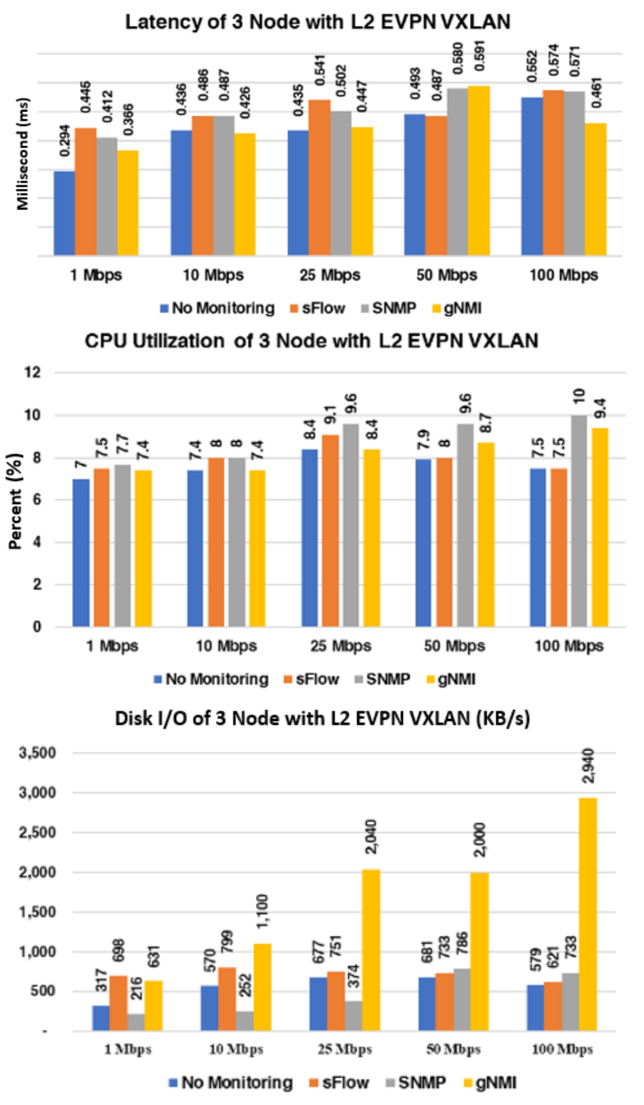

Figure 4. Test result data with TCP data traffic.

TABLE II
OBSERVATION DATA OF gNMI INTERVAL DURATION

Interval Duration	CPU (%)	Memory (GByte)	Disk I/O (KB/s)	Latency (ms)
20 sec	8.5	5.31	1,780	0.460
15 sec	8.9	5.36	1,810	0.462
10 sec	11.7	5.36	2,450	0.489
5 sec	12.8	5.36	4,910	0.555
3 sec	13.4	5.38	6,920	0.544
1 sec	15.0	5.40	13,100	0.516

transmission was investigated. The Prometheus agent interval set in the configuration file needed to be synchronized with the interval duration of the gNMI client to prevent inaccurate information from being stored in the TSDB.

Table II presents the results of the effect of data pulling interval duration on gNMI telemetry performance. The interval duration ranges from 20 s to near real-time. This test employed the UDP packet type with a data rate of 25 Mbps and a test duration of 1 min. The number of subscribed gNMI objects was 65. A data rate of 25 Mbps was selected to assess the characteristics of the gNMI-based monitoring system under ideal conditions, specifically without packet loss.

The data in Table II indicate that a shorter scraping interval configured in gNMI telemetry resulted in higher CPU utilization of the host. Similarly, RAM usage increased as the scraping interval approaches real-time. Disk I/O usage also rose significantly, with values increasing up to six times from the initial scenario with a 20 s scraping interval. This parameter should be considered when planning to deploy a gNMI streaming telemetry-based monitoring system in a production environment. These results may vary when implemented on different Network OS.

TABLE III
OBSERVATION DATA OF SUBSCRIBED OBJECTS

Number of Subscribed Objects	CPU (%)	Memory (GByte)	Disk I/O (KB/s)	Latency (ms)
21	10.0	5.31	4,420	0.420
45	10.7	5.33	5,390	0.483
65	13.6	5.38	6,090	0.474
97	20.4	5.40	10,400	0.590
132	21.9	5.40	19,500	0.578

TABLE IV
OBSERVATION DATA OF gNMI STREAMING TELEMETRY WITH LOG MONITORING

Metrics	gNMI Telemetry	gNMI Telemetry + Syslog
Latency (ms)	0.462	0.465
CPU Utilization (%)	8.9	9.5
Memory (GByte)	5.36	5.43
Disk I/O (KB/s)	1,810	2,410

C. *ANALYSIS OF THE EFFECT OF NUMBER of THE gNMI DATA SUBSCRIPTION OBJECT*

This test scenario aimed to determine the effect of the number of data subscription objects on network device parameters related to the level of usage and traffic within the communication network. Tests were conducted using a 25 Mbps UDP packet type with a test duration of 1 min. The scraping interval was configured to 3 s. Table III presents the test results for this scenario. The subscribed objects in this test refer to the deepest objects contained in the YANG tree hierarchical structure found in Network OS SR Linux.

According to the test observation data presented in Table III, it is evident that as the number of subscribed gNMI objects increased, both the CPU utilization rate and the RAM usage rate on the host also increased. Although the RAM usage did increase, it did not rise as significantly as in the sFlow-based monitoring system. The observation of the Disk I/O utility parameter also indicates an increase in value as the number of subscribed gNMI objects rises. In addition to considering the scraping interval duration, this test indicates that the number of objects to be monitored by the system must also be planned in a production environment to prevent degradation of system and host performance.

D. *PERFORMANCE ANALYSIS OF gNMI INTEGRATION WITH Syslog*

In this scenario, testing was conducted utilizing UDP packet transmission with a data rate of 25 Mbps over a test duration of 2 min, while adhering to a gNMI telemetry sampling interval of 15 s. The tools employed to facilitate the monitoring stack of the system included syslog-ng (v3.38.1), promtail (v2.7.4), and Grafana Loki (v2.7.4). This test pertains to the laboratory outlined by Nokia [23]. The test results are shown in Table IV.

The data presented in Table IV indicates that incorporating gNMI telemetry streaming with syslog did not significantly impact network traffic conditions, as evidenced by the negligible alteration in latency values. RAM usage experienced a slight increase of 70 MB. Noteworthy, however, is the escalation in Disk I/O resources by 33% compared to gNMI telemetry without log monitoring.

This experiment employed three network nodes with a scraping interval set at 15 s and subscribed to a total of 65 objects. Should adjustments be made to these parameters, such as reducing the interval duration and increasing the number of subscribed objects, it is imperative to consider the trade-offs elucidated in the preceding test. Implementing a monitoring system, particularly one reliant on gNMI streaming telemetry in a production environment, necessitated meticulous consideration and planning of host resources. With a greater number of nodes, there existed a likelihood of escalation in the resource utilization of the I/O Disk. In both centralized and distributed monitoring systems, it was customary for implementations to operate on a solitary host alongside other application or microservice implementations. Inadequate planning of the configuration or trade-off parameters of gNMI streaming telemetry could lead to a heightened utilization of Disk I/O resources, thereby impacting the performance of applications or microservices co-existing on the same host as the monitoring system.

IV. CONCLUSION

This study demonstrates that monitoring systems using the gNMI protocol consume 0.5% less CPU compared to other protocols using UDP packets. However, this trend reverses when the packet type is TCP; in this case, SNMP and gNMI-based systems show increasing CPU utilization. Regarding Disk I/O usage, the gNMI-based monitoring system exhibits higher resource consumption than other protocols. The study also examines the integration of gNMI streaming telemetry and log monitoring, revealing a 70 MB increase in memory usage and a 33% rise in Disk I/O resource levels.

This study also indicates a 50% increase in CPU utilization by the gNMI-based system compared to the mode data recorded in the observation. The trade-off analysis conducted in this study reveals that reducing the scraping interval to near real-time and increasing the number of subscribed objects result in higher Disk I/O and CPU utilization. Future research is likely to focus on homogeneous node types and more complex node topologies and these can be tested in real-world scenarios on hardware that supports the gNMI telemetry streaming protocol.

CONFLICTS OF INTEREST

The authors clearly state that they have no conflicts of interest that could potentially affect the results or interpretation of this study. The authors do not possess any pecuniary, or personal interests that might bias an impartial evaluation of the research data or findings. Furthermore, the authors have no affiliations with organizations or companies that could sway the outcome of this study.

AUTHORS' CONTRIBUTIONS

Conceptualization, Fierda Kurniacahya Ariefputra; methodology, Fierda Kurniacahya Ariefputra and Eueung Mulyana; system design, Fierda Kurniacahya Ariefputra; simulation implementation, Fierda Kurniacahya Ariefputra; data collection, Fierda Kurniacahya Ariefputra; writing-original draft, Fierda Kurniacahya Ariefputra; review, Eueung Mulyana.

ACKNOWLEDGMENT

Gratitude is extended to Roman Dodin, the founder and developer of Containerlab and Nokia SR Linux. His invaluable assistance in resolving bugs during the research has been instrumental to the successful completion of this work.

REFERENCES

[1] B. Aggarwal, Q. Xiong, and E. Schroeder-Butterfill, “Impact of the use of the internet on quality of life in older adults: Review of literature,” *Prim. Health Care Res. Develop.*, vol. 21, pp. 1-6, Dec. 2020, doi: 10.1017/S1463423620000584.

[2] J. Manyika, “Big data: The next frontier for innovation, competition, and productivity,” 2011. [Online]. Available: https://personal.utdallas.edu/~muratk/courses/cloud11f_files/MGI-full-report.pdf

[3] E. Permadi (2020) “Industri cloud dan hosting di Indonesia, begini kondisi saat pandemi COVID-19,” [Online], https://sumatra.bisnis.com/read/20200819/534/1280826/industri-cloud-dan-hosting-di-indonesia-begini-kondisi-saat-pandemi-covid-19, access date: 25-Feb-2023.

[4] E.B. Nadales (2023) “Impact of traffic growth on networks and investment needs,” [Online], https://www.telefonica.com/en/ communication-room/blog/impact-of-traffic-growth-on-networks-and-investment-needs/, access date: 5-Jun-2023.

[5] M.Y.B. Rasyiidin and F.A. Murad, “Monitoring server berbasis SNMP menggunakan Cacti pada server lokal,” *J. Ilm. FIFO*, vol. 13, no. 1, pp. 14–23, May 2021, doi: 10.22441/fifo.2021.v13i1.002.

[6] A. Pradana, I.R. Widiasari, and R. Efendi, “Implementasi sistem monitoring jaringan menggunakan Zabbix berbasis SNMP,” *AITI*, vol. 19, no. 2, pp. 248–262, Nov. 2022, doi: 10.24246/aiti.v19i2.248-262.

[7] A. Sgambelluri *et al.*, “Reliable and scalable Kafka-based framework for optical network telemetry,” *J. Opt. Commun. Netw.*, vol. 13, no. 10, pp. E42-E52, Oct. 2021, doi: 10.1364/JOCN.424639.

[8] F. Paolucci, A. Sgambelluri, P. Castoldi, and F. Cugini, “Telemetry solutions in disaggregated optical networks: An experimental view,” *Opt. Fiber Commun. Conf. (OFC) 2021*, 2021, pp. 1–3, doi: 10.1364/OFC.2021.W1G.1.

[9] R. Vilalta *et al.*, “Telemetry-enabled cloud-native transport SDN controller for real-time monitoring of optical transponders using gNMI,” *2020 Eur. Conf. Opt. Commun. (ECOC)*, 2020, pp. 1-4, doi: 10.1109/ECOC48923.2020.9333143.

[10] A. Sgambelluri *et al.*, “Open source implementation of OpenConfig telemetry-enabled NETCONF agent,” *2019 21st Int. Conf. Transparent Opt. Netw. (ICTON)*, 2019, pp. 1–4, doi: 10.1109/ICTON.2019.8840320.

[11] K.S. Mayer *et al.*, “Machine-learning-based soft-failure localization with partial software-defined networking telemetry,” *J. Opt. Commun. Netw.*, vol. 13, no. 10, pp. E122–131, Oct 2021, doi: 10.1364/JOCN.424654.

[12] J.E. Simsarian *et al.*, “Demonstration of cloud-based streaming telemetry processing for optical network monitoring,” *2021 Eur. Conf. Opt. Commun. (ECOC)*, 2021, pp. 1–4, doi: 10.1109/ECOC52684.2021.9605813.

[13] X. Cheng *et al.*, “IntStream: An intent-driven streaming network telemetry framework,” *2021 17th Int. Conf. Netw. Service Manag. (CNSM)*, 2021, pp. 473–481, doi: 10.23919/CNSM52442.2021.9615520.

[14] R.A.K. Fezeu and Z.-L. Zhang, “Anomalous model-driven-telemetry network-stream BGP detection,” *2020 IEEE 28th Int. Conf. Netw. Protoc. (ICNP)*, 2020, pp. 1–6, doi: 10.1109/ICNP49622.2020.9259411.

[15] A. Sadasivarao *et al.*, “Demonstration of extensible threshold-based streaming telemetry for open DWDM analytics and verification,” *Opt. Fiber Commun. Conf. (OFC) 2020*, 2020, pp. 1–3, doi: 10.1364/OFC.2020.M3Z.5.

[16] R.P. Pinto *et al.*, “Packet-optical differentiated survivability implemented by P4 slices and gNMI telemetry,” *2023 Opt. Fiber Commun. Conf. Exhib. (OFC) 2023*, 2023, pp. 1–3, doi: 10.1364/OFC.2023.M1G.3.

[17] Ç. Kurt and O.A. Erdem, “Real-time anomaly detection and mitigation using streaming telemetry in SDN,” *Turkish J. Elect. Eng. Comput. Sci.*, vol. 28, no. 5, pp. 2448–2466, Sep. 2020, doi: 10.3906/elk-1909-112.

[18] I. Ivanov, “Comparing the performance of SNMP to network telemetry streaming with gRPC/GPB,” *53rd Int. Sci. Conf. Inf. Commun. Energy Syst. Technol.*, 2018, pp. 175–178.

[19] E. Pettersson, “A comparison of pull-and push-based network monitoring solutions examining bandwidth and system resource usage,” Undergraduate thesis, KTH Royal Institute of Technology, Stockholm, Swedia, 2021.

[20] B. Buresh *et al*. *A Modern, Open, and Scalable Fabric VXLAN EVPN*. (2016). Access date: 7-Mar-2023. [Online]. Available: https://www.cisco.com/c/dam/en/us/td/docs/switches/datacenter/nexus9000/sw/vxlan_evpn/VXLAN_EVPN.pdf

[21] H. Song *et al.* (2022) “Network Telemetry Framework RFC 9232,” [Online], https://datatracker.ietf.org/doc/html/rfc9232, access date: 10-Mar-2023.

[22] M. Korshunov. Streaming telemetry: Considerations & challenges [Online]. Available: https://ripe78.ripe.net/presentations/26-ripe78_Korshunov_Streaming_Telemetry_consideration_and_challenges_final.pdf

[23] R. Dodin, B. Claeys, and M. Vahlenkamp, “Nokia SR Linux Streaming Telemetry Lab,” [Online], https://github.com/srl-labs/srl-telemetry-lab, access date: 10-Mar-2023.