International Journal of Computer and Organization Trends
ISSN: 2249 – 2593 / https://doi.org/10.14445/22492593/IJCOT-V15I1P304

Volume 15 Issue 1, 33-40, Jan-Apr 2025
© 2025 Seventh Sense Research Group®

Original Article

A Multi-Agent Monitoring System for Computer Networks

Amwayi Harrison[1], Abraham Mutua[2]

[1,2]*Department, Computing and Information Technology, Kenyatta University, Kenya.*

[1]*Corresponding Author : harryamwayi@gmal.com*

Received: 09 January 2025 Revised: 13 February 2025 Accepted: 05 March 2025 Published: 19 March 2025

Abstract - *In today's business landscape, reliable network infrastructure is essential for uninterrupted operations. As networks increase in complexity, effective monitoring systems become crucial to ensure device performance and availability. This study presents a multi-agent-based system for monitoring computer network devices, utilizing autonomous agents to gather data through the Simple Network Management Protocol (SNMP). The proposed solution addresses scalability and network congestion challenges often seen in centralized monitoring systems by leveraging Apache Kafka to distribute and manage monitoring data efficiently while autonomous agents handle data collection, thereby reducing latency and minimizing network load. The research first examines existing multi-agent applications that leverage SNMP agents, identifying their limitations before introducing a novel model that integrates SNMP agents with Apache Kafka to facilitate scalable data ingestion. The system's performance was tested in a simulated environment, and significant enhancements in scalability and efficiency were observed for real-time monitoring. The findings conclude that combining multi-agent systems with Apache Kafka offers a robust framework for efficient, real-time network monitoring, with improved scalability and reduced latency compared to traditional centralized models.***

Keywords - *Network Monitoring, Multi-Agent Systems, Apache Kafka, Simple Network Management Protocol (SNMP), Scalability.*

1. Introduction

The rapid expansion of network technologies has significantly increased the complexity of managing modern networks. Efficient network management systems are essential for optimizing resource utilization, maintaining reliability and ensuring high performance [2]. As networks grow in scale and dynamism, traditional approaches to monitoring and troubleshooting face increasing limitations, making real-time decision-making more challenging [8], [13]. Network monitoring is critical in overcoming these challenges by providing administrators with real-time insights into traffic patterns, device utilization and key performance metrics necessary for proactive management [11].

Modern network monitoring systems have evolved to incorporate automation, intelligent analytics, and scalable architectures to keep pace with industry demands, improving overall network control and responsiveness [3]. Among these frameworks, the Simple Network Management Protocol (SNMP) remains one of the most widely used standards for monitoring and managing network devices. SNMP enables administrators to retrieve device statistics, modify configurations and analyse network performance [6]. Over the years, multiple versions of SNMP have been developed, with SNMPv3 offering enhanced security and reliability [5], [12].

However, despite its widespread adoption, traditional SNMP-based monitoring architectures exhibit significant scalability challenges in large-scale environments. Conventional implementations rely on centralized Network Management Stations (NMS) that collect and process SNMP data from all monitored devices. As the number of network devices increases, centralized architectures experience traffic congestion and increased latency and processing bottlenecks, ultimately limiting real-time analytics and decision-making capabilities. This issue is particularly problematic in dynamic and large-scale networks where rapid data processing is critical.

Existing research has primarily focused on improving SNMP security, optimizing query efficiency and enhancing protocol performance [4], [7]. However, limited attention has been given to addressing the scalability bottlenecks inherent in centralized SNMP-based monitoring systems. While some solutions advocate vertical scalability—enhancing hardware resources such as CPU, memory and storage—this approach is constrained by physical limitations and escalating costs. In contrast, horizontal scalability, which involves distributing workloads across multiple interconnected nodes, offers a more flexible and sustainable solution for large-scale network monitoring.

This research addresses the critical gap in scalable SNMP-based monitoring by proposing a distributed architecture. The proposed system integrates SNMP with Apache Kafka and MySQL to facilitate scalable and efficient SNMP data collection and processing by distributing monitoring tasks across multiple nodes; this approach mitigates the limitations of centralized architectures, enabling improved performance, reduced latency and enhanced fault tolerance.

The remainder of this paper is organized as follows. Section II introduces the Simple Network Management Protocol (SNMP), its evolution and its role in network monitoring. It reviews related works on SNMP-based monitoring, focusing on scalability challenges and advancements in the field. Section III details the tools and methodology for implementing a scalable SNMP monitoring system with Apache Kafka, MySQL, and a multi-agent architecture. Section IV presents the results, including performance evaluations and scalability analysis. Section V concludes the study and discusses potential enhancements for optimizing distributed SNMP monitoring.

2. Simple Network Management Protocol (SNMP)

The first version of the Simple Network Management Protocol (SNMP) was first proposed in RFC1157 in May 1990 [16]. Today, SNMP agents are embedded in nearly all network-enabled devices [10]. Designed for minimal agent complexity, extensibility, and independence from specific hosts or gateways, SNMP offers key advantages such as compatibility, simplicity and a small agent footprint [1]. As part of the TCP/IP protocol suite, SNMP is widely used for monitoring network devices and data center equipment. In a networked environment, each device with an SNMP agent communicates status updates, enabling efficient monitoring and management.

An SNMP monitoring system consists of three key elements [9] shown in Fig.1:

- Devices/Agent station – These are network-enabled hardware components that require monitoring. An SNMP agent comes pre-installed on these devices to facilitate communication.
- Agents – These are software programs running on the monitored devices. Agents collect and store data about system performance, status and resource usage and store it in a management information base (MIB). They respond to SNMP requests from the network management station and send alerts when certain thresholds or events occur.
- Management station – Also known as the SNMP manager, this is a centralized system that communicates with SNMP agents to collect and analyze network data. It processes the received information, generates reports and provides real-time monitoring and alerting capabilities.

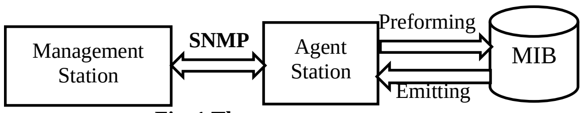

Fig. 1 The manager-agent model

SNMP operates at Layer 7 of the OSI model and uses UDP port 161 for communication. Since it is primarily used for monitoring, it does not require acknowledgements, making UDP a suitable transport protocol. The basic structure of an SNMP Protocol Data Unit (PDU) includes an IP and UDP header, followed by the SNMP version, community string, request ID, error status, error-index, and variable bindings. Fig 2 below shows the basic structure of an SNMP data PDU.

IP header	UDP header	version	Community	PDU-type	request-ID	error-status	error-index	variable bindings

Fig. 2 Basic structure of the SNMP data PDU

There are three main versions of the SNMP protocol. Version 1 ([RFC 1065, 1066, 1067, 1156]) had security and authentication vulnerabilities, which were addressed in Version 2 ([RFC 1213]). However, Versions 1 and 2 are not interoperable due to differences in message formats and protocol operations. Version 3 ([RFC 3411, 3418]) introduced remote configuration enhancements and improved security mechanisms.

2.1. Network Monitoring

In today's interconnected world, the need for efficient, real-time network monitoring tools has become paramount. Among the most commonly used tools in the industry are Nagios, Zabbix, SNMP MIB Browser Android Tool and Cacti.

2.1.1. Nagios

Nagios is an open-source software used to monitor the availability of network devices and services. It functions as a fault-monitoring tool that uses plugins to track various network services such as HTTP, DNS, PING, and SNMP [14]. Nagios is known for its flexibility and customizability, allowing users to monitor both network services (e.g., HTTP, SMTP) and host resources (e.g., CPU, memory). It also supports SNMP-WALK for retrieving object identifiers (OIDs), making it effective for querying and monitoring device-specific information from device Management Information Bases (MIBs).

One of Nagios' key advantages is its open-source version, Nagios Core, which makes it a cost-effective solution for organizations with limited budgets. Its extensive plugin library also enables users to customize monitoring based on their specific needs, making it suitable for enterprise-grade networks. However, Nagios has a steep learning curve, particularly for beginners, and its configuration can be complex, especially in large-scale deployments.

From an SNMP data retrieval perspective, Nagios is well-suited for small to medium-sized networks but may face performance challenges in large-scale environments with numerous Object Identifiers (OIDs) or devices. Its use of periodic SNMP-WALKs allows efficient collection of MIB data but does not support real-time monitoring. For real-time SNMP event handling, integrating SNMP traps is recommended.

2.1.2. Zabbix

Zabbix is an open-source tool that provides network and application monitoring with support for SNMP and advanced visualization features like graphs and maps. Its SNMP capabilities include SNMP-WALK for bulk OID retrieval, SNMP GET for targeted queries, and SNMP TRAPs for real-time event monitoring [15].

A major advantage of Zabbix is that it is completely free and open-source, making it accessible to organizations of all sizes. It also benefits from active community support and extensive documentation. However, the initial setup can be time-consuming.

Zabbix is highly scalable when properly configured, using techniques such as distributed monitoring with proxies and database optimization. While SNMP-WALK enables efficient MIB data retrieval, large-scale SNMP queries may cause performance bottlenecks if the system is not tuned effectively

2.1.3. SNMP MIB Browser Android Tool

The SNMP MIB Browser Android Tool is a lightweight, portable SNMP monitoring solution. It supports SNMP GET, SET, and WALK operations directly from a mobile device, making it ideal for quick, on-the-go troubleshooting [17]. The tool also enables interactive browsing of MIBs for device-specific OIDs, and its ability to perform SNMP-WALK provides rapid access to multiple OIDs.

Its primary advantages lie in its convenience and ease of use. It requires no additional hardware or complex setup, and its user-friendly interface makes it accessible to many users. However, its limitations include minimal scalability and lack of historical data storage.

For SNMP MIB data retrieval, this tool is highly effective for single-device troubleshooting but lacks automation and scalability for enterprise environments. Since this tool relies on manual SNMP-WALK operations, it is unsuitable for large-scale or continuous monitoring scenarios requiring automated polling and alerting.

2.1.4. Cacti

Cacti is an open-source network monitoring tool specializing in graphing and visualization. It gathers data from devices using SNMP and supports SNMP-WALK for retrieving multiple OIDs simultaneously [9]. Leveraging RRDTool, Cacti creates detailed and customizable graphs that help visualize network performance metrics over time.

One of Cacti's major strengths is its ability to create highly customizable graphs and templates, making it an excellent choice for organizations where visualization is a priority. However, its focus on graphing limits its functionality as a comprehensive monitoring tool, lacking real-time alerting and event-driven monitoring capabilities.

While Cacti supports SNMP-WALK, its setup requires manual configuration, which can be time-consuming in large networks. Regarding SNMP MIB data retrieval, Cacti is ideal for environments that prioritize visualization but lack the automation of more feature-rich tools like Zabbix or Nagios.

Despite the strengths of these tools, their limitations in SNMP MIB data retrieval highlight the need for a more scalable and efficient monitoring solution. While Nagios and Zabbix offer flexibility, they can become resource-intensive when handling large-scale SNMP queries. Cacti's primary focus on visualization makes it less effective as a comprehensive monitoring solution, and the SNMP MIB Browser Android Tool, while convenient, is inherently limited to small-scale use cases. These challenges emphasize the need for optimized SNMP data retrieval methods that scale in large network monitoring scenarios.

3. Methodology

Building on the highlighted challenges of applications that utilize SNMP for computer network monitoring, a new model is proposed that leverages Apache Kafka to optimize data collection times through a distributed architecture to overcome the traditional bottlenecks. This model employs a multi-agent design approach to enhance scalability and efficiency.

3.1. System Architecture

The system was designed to handle large-scale, distributed SNMP polling across multiple hosts while ensuring real-time data streaming and storage. It leverages Apache Kafka as the backbone for message communication between producers (polling agents) and consumers (data processors), enabling seamless data flow. By incorporating multithreading, the system maximizes the efficiency of SNMP polling, while MySQL provides robust and structured data storage.

To further optimize performance, the system employs feature selection in its polling strategy, focusing on specific high-priority MIB objects rather than relying on bulk operations like snmpwalk. This targeted approach reduces network bandwidth consumption and optimizes storage by collecting only essential data.

Building on this foundation, the adopted approach follows a four-layer design where the SNMP manager sends poll requests to end devices across the network. Each layer performs a specific function to achieve the overall objective of efficiently retrieving Object Identifier (OID) parameters from these monitored devices. The structure of these layers is illustrated in Figure 3.

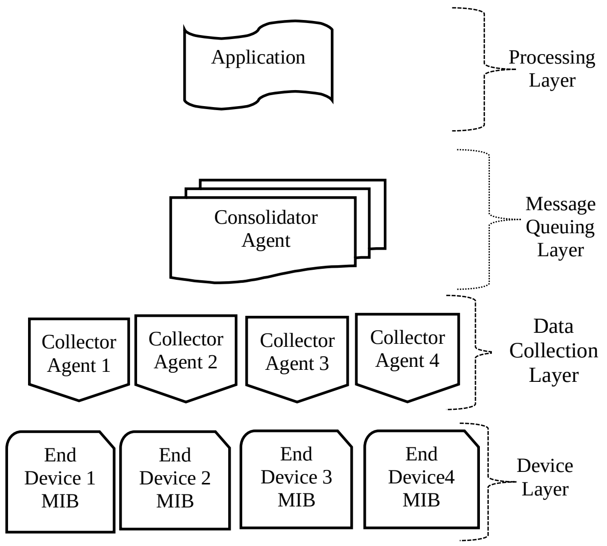

Fig. 3 multi-agent system architecture

3.1.1. Device Layer

This layer consists of SNMP-enabled end devices that expose their metrics and statuses through Object Identifiers (OIDs), providing a structured way to access specific data points.

3.1.2. Data Collection Layer

The data collection layer is responsible for gathering SNMP data from the end devices. Central to this layer is the SNMP library and a Python script that performs SNMP GET and GETNEXT operations to query various metrics from the devices.

3.1.3. Message Queuing Layer

The message queuing layer serves as an intermediary that decouples data collection from data processing, enhancing the overall system's flexibility and scalability. Utilizing Kafka, this layer efficiently manages the flow of data collected (using topics) from the end devices.

3.1.4. Processing Layer

The processing layer is responsible for analysing and transforming the data consumed, enabling valuable insights and actionable intelligence. Central to this layer is the consumer, which retrieves messages from the specified Kafka topics.

3.2. System Design

The system design follows a producer-consumer model, with Apache Kafka acting as the communication layer between the SNMP polling process and the database insertion process. The producer agents poll network devices using SNMP and publish collected OID parameters to Kafka topics, enabling real-time data streaming. Kafka serves as a buffer and a message broker, ensuring scalability and fault tolerance.

The consumer agents subscribe to these topics, process the SNMP data and insert structured records into a MySQL table, enabling efficient storage and retrieval. This decoupled architecture optimizes performance by preventing direct database dependencies, reducing network congestion and improving the scalability of SNMP-based monitoring.

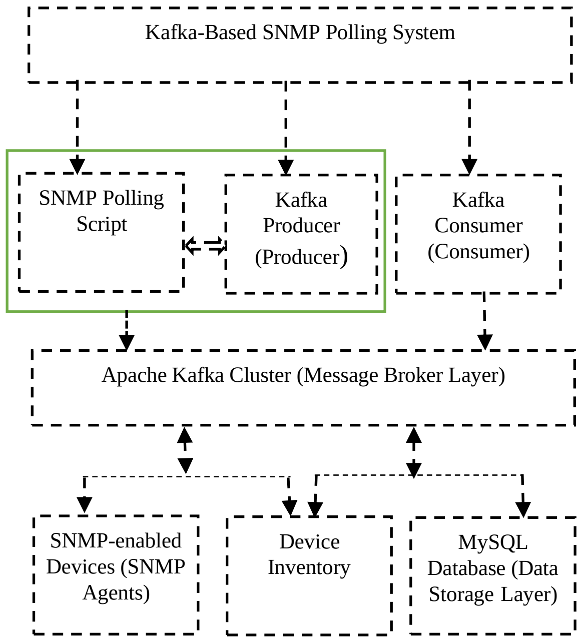
Kafka-Based SNMP Polling System
SNMP Polling Script
Kafka Producer (Producer)
Kafka Consumer (Consumer)
Apache Kafka Cluster (Message Broker Layer)
SNMP-enabled Devices (SNMP Agents)
Device Inventory
MySQL Database (Data Storage Layer)

Fig. 4 high-level design breakdown

3.2.1. SNMP Polling Script and Device Inventory

The SNMP polling system serves as the core data collection engine, efficiently retrieving SNMP data from network devices. It operates in a continuous loop, where each thread polls a specific host and MIB OID using the

EasySNMP library. The system calculates the turnaround time for each poll to assess network performance. The collected data—comprising the host, MIB OID, value, and turnaround time—is serialized into JSON and published to an Apache Kafka topic. Leveraging multithreading, the system enables concurrent polling of multiple devices, ensuring scalability and optimized data acquisition across large networks.

A device inventory file defines hosts, SNMP communities, MIBs, and polling frequencies, enabling a modular and adaptable configuration approach. This design offers:

- Dynamic Polling Configuration: Devices can be added or removed by modifying the inventory file without altering the core script.
- Host-Specific Polling: Each device can be assigned a unique set of MIBs and polling frequencies, allowing for targeted and flexible monitoring.

This design enhances the efficiency, scalability, and adaptability of SNMP-based network monitoring, making it suitable for large-scale distributed environments.

Algorithm 1: Producer

```
Require: Inventory file containing <Host, Frequency, Community, Version, MIBs>
    1. Load Inventory File
        o Read device details from inventory.csv.
        o Extract Host, Polling Frequency, SNMP Community, SNMP Version, and MIBs.
    2. For Each Device in Inventory:
        o Create a polling thread for each MIB
    3. Polling Thread Execution (Parallel for Each MIB):
        o Establish an SNMP session.
        o Loop:
            ▪ Record Start Time.
            ▪ Query MIB using SNMP GET.
            ▪ Capture response and timestamps (Date, Time).
            ▪ Compute Turnaround Time.
            ▪ Serialize data as JSON.
            ▪ Publish to Apache Kafka (snmp_thread topic).
            ▪ Sleep for Polling Frequency.
    4. Handle Errors:
        o Log and retry on failure to avoid overwhelming the SNMP agent.
Return: Real-time SNMP data stream sent to Kafka for further processing.
```

3.2.2. Apache Kafka as a Message Broker

Apache Kafka serves as a distributed message broker, facilitating scalable and fault-tolerant communication between the SNMP polling script and the database insertion process. By decoupling these components, Kafka ensures efficient data flow and high availability in large-scale SNMP monitoring systems.

Key Roles of Kafka in the System are:

- Reliable Message Distribution: Kafka transmits SNMP poll results from producers to consumers via a dedicated topic, ensuring data integrity.
- Scalability: The system accommodates increased data loads by dynamically scaling consumer instances for parallel processing.
- Fault Tolerance: Kafka's message retention and offset tracking mechanisms enable data recovery in the event of failures, preventing information loss.

All polling results are published to a designated Kafka topic in JSON format. Multiple consumers can subscribe to this topic, enabling real-time processing and storage.

3.2.3. Kafka Consumer and MySQL Database Integration

The system employs an Apache Kafka consumer that seamlessly integrates with a MySQL database to ensure efficient processing and storage of SNMP polling results. This setup enables real-time data ingestion.

The Kafka consumer continuously listens to a designated topic, retrieving SNMP poll results as they arrive. Each message is processed through the following steps:

- Message Decoding: The consumer parses the JSON message, extracting key details such as the host, MIB OID, value, and turnaround time.
- Data Validation: Before insertion, the data undergoes a completeness check to ensure all required fields are present and valid.
- Database Insertion: Once validated, the data is stored in a MySQL database, where turnaround time serves as a crucial metric for analyzing network performance.

The MySQL database acts as the system's persistent storage layer, organizing SNMP polling results into a MySQL table. This structured format enables efficient data retrieval and analysis, capturing essential parameters about the SNMP host being monitored.

Algorithm 2: Consumer

```
Require: Kafka Broker, MySQL Database, SNMP Polling Data

  Initialize Kafka Consumer
  1.1. Set up consumer configurations (Kafka broker, consumer group, offset reset).
  1.2. Subscribe to the snmp_thread topic.
  Consume Messages Continuously
  2.1. Poll Kafka for new messages with a 1-second timeout.
  2.2. If no message is received, continue polling.
  2.3. If an error occurs, handle and log the error.

 Process Received Messages
  3.1. Decode JSON message to extract SNMP poll data.
```

```
3.2. Validate required fields: <host, mib, value, turnaround_time, poll_date, poll_time>.
3.3. If essential fields are missing, log an error and discard the message.

Insert Data into MySQL
4.1. Establish a connection to the MySQL database.
4.2. Execute an INSERT query to store the SNMP poll
4.3. Commit the transaction and close the connection.

Handle Exceptions
5.1. If JSON decoding fails, log and continue processing.
5.2. If database insertion fails, log the error and retry if necessary.

Repeat Process
6.1. Continue consuming messages indefinitely.
6.2. Gracefully close the Kafka consumer upon termination.

Return: SNMP poll results are stored in MySQL for further analysis and reporting.
```

3.3. System Simulation

The system was deployed and tested in a controlled environment designed to simulate real-world SNMP polling scenarios to ensure reproducibility. The software stack included Apache Kafka (latest version) for real-time data streaming, MySQL 8.0 for structured data storage, and Python 3.9 with the EasySNMP and Kafka-Python libraries to facilitate SNMP polling and message handling.

The testbed comprised a mix of physical network switches, desktop computers, and virtual Docker containers, collectively simulating a diverse set of SNMP-enabled network devices. The network topology was structured with one SNMP manager overseeing 54 SNMP-enabled hosts, ensuring a representative distribution of polling tasks. Both SNMP v2c and SNMP v3 configurations were tested to accommodate different SNMP implementations and compatibility requirements.

3.4. System Validation

To ensure the reliability, stability and efficiency of the SNMP monitoring system, a series of rigorous performance tests were conducted. These evaluations aimed to identify potential bottlenecks, validate system resilience under varying load conditions and verify long-term sustainability. The primary testing methodologies employed included Load Testing, Simulated high-traffic scenarios to evaluate the system's ability to handle concurrent SNMP poll requests without degradation in performance, Spike Testing, Assessing the system's response to sudden surges in polling activity to ensure that the system could handle abrupt increases in demand without failure, and Soak Testing; Verified long-term reliability by running continuous polling over extended periods, demonstrating that the system maintained stability and avoided issues like memory leaks or resource exhaustion. Each of these tests provided valuable insights into the system's behavior under different stress conditions and helped refine its performance to meet operational requirements.

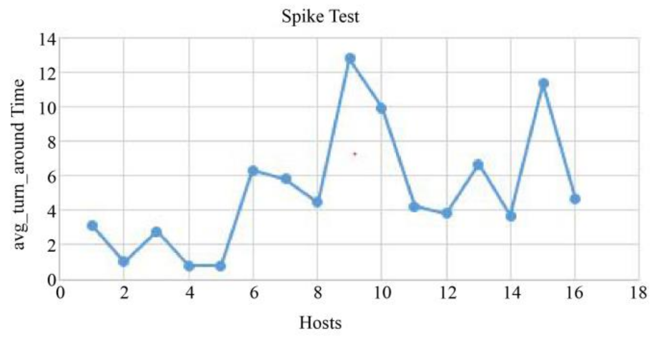

Fig. 5 Spike test for 16 hosts

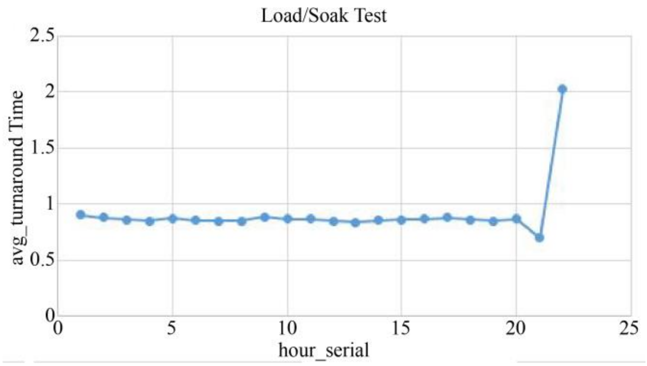

Fig. 6 Load and soak Testing results using 54 hosts over 21 hours

4. Results and Discussion

A subset of the SNMP polling data is stored in a MySQL database table. The data includes:

- Host: The IP address or hostname of the device being polled.
- MIB: The SNMP object identifier is being polled.
- Value: The returned value from the SNMP query.
- Turnaround Time: The time taken to retrieve the SNMP data in milliseconds.

Table 1. Sample data from MySQL

Host	MIB	Value	Turnaround time
Host-1	sysName.0	Server-A	1.19019
Host-1	sysLocation.0	Data Center	1.02639
Host-2	sysName.0	Server-B	1.86555
Host-2	hrMemorySize.0	33518868	1.92642
Host-2	sysLocation.0	IT-Block	1.42853
Host-3	sysName.0	SNMP host	1.58143

The turnaround time is an important metric that measures the system's polling performance, a critical factor for monitoring large-scale networks.

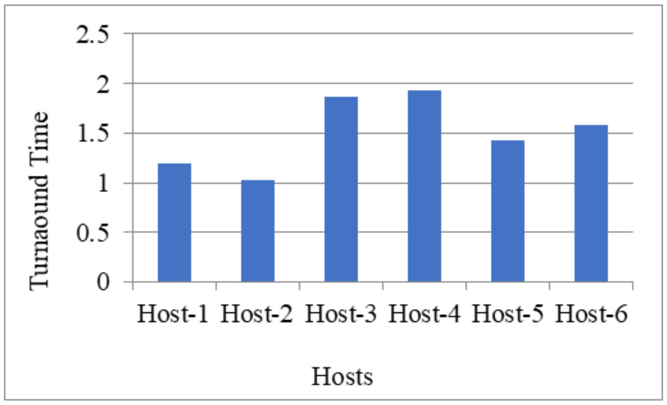

Fig. 7 The average turnaround time for SNMP polling across different hosts

4.1. Latency Breakdown and Polling Rate Analysis

From Table 1 above, the latency breakdown for Host-1 shows that the sysName.0 MIB has a turnaround time of 1.19019 seconds, while sysLocation.0 has a slightly lower turnaround time of 1.02639 seconds.

These variations may be attributed to differences in network latency or the complexity of MIB data retrieval.

Given a polling interval of 5 seconds, the polling rate per host is calculated as: -
Polling Rate (per host) = 1/Polling Interval(seconds) = 1/5
= 0.2 polls/second.

With 54 hosts in the experiment, the total polling rate is:
Total Polling Rate = Polling Rate per Host × Number of Hosts = 0.2 × 54 = 10.8 polls/second.
This indicates that the system handles 10.8 SNMP requests per second across all monitored hosts.

4.2. Queue Length Estimation Using Little's Law

To assess system performance, Little's Law is applied,

$$L = \lambda \times W$$

where:

L = average queue length
λ = total polling rate (10.8 polls/second)
W = average turnaround time (1.503 seconds)

L =1 0.8×1.503 =16.2 messages

This result suggests that, on average, 16.2 messages are in the queue at any given time. The queue length aligns with observed turnaround times, helping to identify thresholds for performance tuning. If queue lengths grow excessively, it could indicate system bottlenecks requiring optimization.

4.3. Decision Theory for Resource Allocation

Decision Theory can guide resource allocation based on data criticality to enhance performance. For example, if hrMemorySize.0 for Host-2 is critical, its higher turnaround time (1.92642 seconds) suggests that prioritizing its processing would be beneficial, hence employing optimization strategies that may include:

- Dedicating Kafka partitions for critical MIBs
- Additional consumer threads for high-priority SNMP data.
- Load balancing to minimize processing delays.

4.4. Limitations of the Study

While the system effectively monitors and processes SNMP data, certain limitations may impact its real-world applicability:

- Fault Tolerance: Although Kafka ensures message reliability, polling agents lack automatic failover mechanisms. Agent failures could disrupt data collection, requiring future enhancements like redundant agents.
- Testing Environment: The system was evaluated in a controlled setup, lacking real-world network challenges such as latency fluctuations and intermittent connectivity. Further testing in diverse production environments is needed to assess scalability and robustness.

5. Conclusion

This study successfully demonstrated the potential of multi-agent-based systems in overcoming the limitations of traditional network monitoring solutions. By leveraging a decentralized and scalable architecture, the developed system provides an efficient framework for managing the complexity of modern network infrastructures. The use of Apache Kafka as a message broker combined with a multithreaded SNMP polling mechanism ensures real-time data collection, fault tolerance and seamless integration with storage and analytics platforms.

While the system has proven effective, there are several areas for future research to enhance its performance and adaptability. One key area is enhanced fault tolerance, where future work could focus on implementing automatic failover and agent recovery mechanisms. This would improve system resilience, ensuring continuous operation even in the face of failures.

Another promising direction is the integration of advanced analytics, such as machine learning or AI-based models, to predict failures, detect anomalies in network behavior and enable proactive monitoring. This would minimize human intervention while improving the system's ability to handle large-scale networks efficiently.

Overall, this research lays a strong foundation for future advancements in network monitoring, highlighting the potential of scalable, autonomous systems to meet the growing demands of modern network environments.

Funding Statement

The author funded this research and publication independently without financial support from any external organization or grant.

References

[1] Leena Aarikka-Stenroos, and Paavo Ritala, "Network Management in the Era of Ecosystems: Systematic Review and Management Framework," *Industrial Marketing Management*, vol. 67, pp. 23-36, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[2] Sisay Tadesse Arzo et al., "Multi-Agent-Based Traffic Prediction and Traffic Classification for Autonomic Network Management Systems for Future Networks," *Future Internet*, vol. 14, no. 8, pp. 1-23, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Sisay Tadesse Arzo et al., "Multi-Agent Based Autonomic Network Management Architecture," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3595-3618, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Bilal Zaka, and Christian Safran, "Emerging Web-Based Learning Systems and Scalability Issues," *International Conference on Computer Science and Software Engineering*, Wuhan, China, pp. 889-892, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[5] D. Harrington, R. Presuhn, and B. Wijnen, "An architecture for Describing Simple Network Management Protocol (SNMP) management frameworks (RFC 3411)," *Internet Engineering Task Force (IETF)*, 2002. [Google Scholar] [Publisher Link]

[6] Hari T.S. Narayanan, Geetha Ilangovan, and Sumitra Narayanan, "Feasibility of SNMP OID Compression," *Journal of King Saud University - Computer and Information Sciences*, vol. 25, no. 1, pp. 35-42, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[7] Tobias Hossfeld, Poul E. Heegaard, and Wolfgang Kellerer, "Comparing the Scalability of Communication Networks and Systems," *IEEE Access*, vol. 11, pp. 101474-101497, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[8] Oliver Michel, and Eric Keller, "SDN in Wide-Area Networks: A Survey," *Proceedings of the 2017 Fourth International Conference on Software Defined Systems (SDS)*, Valencia, Spain, pp. 37-42, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[9] Ronald Paucar Curasma, and Herminio Paucar Curasma, "Assessment and Proposal of a Network Monitoring System Based on Free Software," *2020 IEEE Engineering International Research Conference (EIRCON)*, Lima, Peru, pp. 1-4, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] Pang-Wei Tsai et al., "Network Monitoring in Software-Defined Networking: A Review," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3958-3969, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[11] Fung Po Tso, Simon Jouet, and Dimitrios P. Pezaros, "Network and Server Resource Management Strategies for Data Centre Infrastructures: A Survey," *Computer Networks*, vol. 106, pp. 209-225, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[12] S. Waldbusser, "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP) (RFC 3418)," *Internet Engineering Task Force (IETF)*, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[13] J. West, Data Communication and Computer Networks: A Business User's Approach, Cengage Learning, 2022. [Online]. Available: https://www.tamuct.edu/syllabi/docs/2023_Summer/20230660157.pdf

[14] Wojciech Kocjan, *Learning Nagios 3.0. Birmingham*, UK: Packet Publishing Ltd., 2008. [Google Scholar] [Publisher Link]

[15] Yang Guo Shan et al., "Research on Monitoring of Information Equipment Based on Zabbix for Power Supply Company," *3rd International Conference on Applied Machine Learning*, Changsha, China, pp. 487-491, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] J. Case et al., "A Simple Network Management Protocol, Request for Comments (RFC) 1157," *Internet Engineering Task Force (IETF)*, 1990. [Google Scholar] [Publisher Link]

[17] Fernando Hidalgo, and Eric Gamess, "Integrating Android Devices into Network Management Systems based on SNMP," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 5, pp. 1-8, 2014. [CrossRef] [Google Scholar] [Publisher Link]