International Journal of Informatics and Computation (IJICOM)
Vol. 8, No. 1, 2026
ISSN: 2685-8711, E-ISSN: 2714-5263 DOI: 10.35842/ijicom

Evaluating SNMP Protocol for Monitoring and Restoring Device Access using RAD Method

Andrea Marie Baikole[1], Joko Purwadi[2], Gani Indriyanta[3]

Abstract
This study proposes a hybrid network monitoring system that integrates Simple Network Management Protocol (SNMP), Neighbor Discovery, and MAC-Telnet mechanisms to improve device monitoring and recovery capabilities. Experimental evaluation was conducted through six testing scenarios, including network scanning, device discovery, interface monitoring, and connectivity recovery. The system successfully discovered 36 network devices in less than 2 minutes, demonstrating a significant improvement compared to conventional sequential scanning methods that required approximately 45 minutes, representing about a 96% reduction in discovery time. The monitoring module also demonstrated high responsiveness in detecting network changes. Interface status monitoring consistently identified physical port state changes within 10–15 seconds, enabling faster detection of connectivity issues and improving real-time network visibility. These results indicate that the proposed system provides efficient monitoring performance suitable for network laboratory environments. Furthermore, the hybrid discovery and self-healing mechanism enabled device recovery even when Layer-3 connectivity was lost due to invalid IP configurations. By utilizing Neighbor Discovery and MAC-Telnet at Layer-2, the system allowed administrators to remotely restore device configurations without physical intervention. Across all six experimental scenarios, the proposed architecture achieved a 100% operational success rate, confirming its reliability and effectiveness for resilient network monitoring and management.

Keywords:
SNMP Protocol, Hybrid Discovery, Monitoring, RAD

1. Introduction

Modern computer networks play a critical role in supporting organizational operations, cloud services, and distributed computing environments. As network infrastructures grow in scale and complexity, administrators require reliable mechanisms to monitor device performance, detect failures, and maintain continuous connectivity. Network monitoring systems therefore become essential tools for maintaining network availability and operational stability. Among various monitoring approaches, the Simple Network Management Protocol (SNMP) remains one of the most widely adopted standards because it enables centralized management of heterogeneous network devices such as routers, switches, and servers. SNMP provides a structured method for retrieving management information from devices through Management Information Base (MIB) objects, allowing administrators to monitor performance indicators and detect abnormal conditions in real time. This capability supports proactive network management and reduces the risk of prolonged service disruption. However, despite its widespread use, many organizations still

Corresponding Author: Andrea Marie Baikole, Universitas Kristen Duta Wacana, Indonesia (andrea.marie@ti.ukdw.ac.id)
1 Andrea Marie Baikole, Faculty of Information Technology, Universitas Kristen Duta Wacana, Indonesia (andrea.marie@ti.ukdw.ac.id)
2 Joko Purwadi, Faculty of Information Technology, Universitas Kristen Duta Wacana, Indonesia (jokop@staff.ukdw.ac.id)
3 Gani Indriyanta, Faculty of Information Technology, Universitas Kristen Duta Wacana, Indonesia (ganind@staff.ukdw.ac.id)

face challenges in integrating SNMP monitoring with automated device access restoration mechanisms, particularly in dynamic network environments. [1], [3], [4], [19]

Network administrators often rely on monitoring tools to detect failures or degraded performance across network devices. Traditional monitoring systems typically focus on passive observation, meaning they only provide alerts when a problem occurs without automatically initiating corrective actions. This limitation creates operational inefficiencies because administrators must manually investigate and restore device connectivity after receiving notifications. Several studies emphasize that effective network management requires not only monitoring capabilities but also mechanisms that support rapid response and automated intervention when device access becomes unavailable. The integration of monitoring and recovery mechanisms therefore becomes an important research topic in modern network management systems. By combining monitoring data with automated restoration procedures, administrators can significantly reduce downtime and improve network resilience. Nevertheless, many existing implementations still treat monitoring and recovery as separate operational processes. [3], [4], [20]

The SNMP protocol provides a standardized communication mechanism that allows network management systems to collect device information and monitor operational parameters. SNMP agents embedded in network devices periodically expose performance metrics, configuration states, and event notifications through defined MIB structures. Network management systems query these parameters using SNMP operations such as GET, SET, and TRAP to obtain real-time status information. Previous studies demonstrate that SNMP-based monitoring systems effectively detect network anomalies, traffic fluctuations, and device failures. Furthermore, researchers highlight the scalability of SNMP for monitoring large distributed networks because it supports lightweight communication and standardized management structures. Despite these advantages, SNMP implementations often focus solely on data retrieval and visualization rather than active restoration of network accessibility. This limitation creates a gap between monitoring functionality and operational recovery processes. [5], [8], [19]

In recent years, researchers have explored the application of SNMP monitoring in various environments, including enterprise networks, cloud infrastructures, and Internet of Things (IoT) ecosystems. The increasing number of connected devices introduces new challenges in monitoring system performance and maintaining reliable network connectivity. Studies on IoT monitoring architectures emphasize that traditional monitoring systems must evolve to support distributed device management and automated fault detection. SNMP remains relevant in these environments because it provides a universal management interface that can interact with diverse hardware platforms. However, researchers also identify that monitoring systems must incorporate intelligent response mechanisms capable of handling device failures and restoring connectivity automatically. Without such mechanisms, administrators still depend on manual intervention, which may delay service restoration and negatively impact system reliability. [8], [19]

Another critical challenge in network management lies in the accessibility of network devices during operational disruptions. When network devices become unreachable due to configuration errors, network congestion, or hardware faults, administrators must quickly restore access to prevent cascading failures across the network infrastructure. Remote access mechanisms such as VPN tunnels or secure management interfaces are often implemented to maintain administrative control over distributed devices. However, these mechanisms alone do not guarantee automatic restoration of device accessibility when failures occur. Effective monitoring systems must therefore integrate device status detection with automated recovery procedures. Such integration ensures that the system can not only identify connectivity issues but also attempt restoration actions based on predefined operational rules. [4], [9]

Developing integrated monitoring and restoration systems requires an efficient software development approach that allows rapid prototyping and iterative improvement. Traditional development models often involve long design cycles that may delay the deployment of operational monitoring solutions. In contrast, the Rapid Application Development (RAD) method emphasizes iterative development, user feedback, and rapid prototyping to accelerate system implementation. RAD allows developers to construct functional prototypes quickly and refine system features based on practical operational requirements. Several studies demonstrate that RAD-based development significantly reduces development time while maintaining system quality and user satisfaction. For network monitoring applications, this approach enables developers to rapidly integrate monitoring modules, data processing mechanisms, and user interfaces within a unified management system. [11], [12], [13]

The integration of SNMP monitoring capabilities with a RAD-based development approach offers several advantages for building practical network management systems. By leveraging SNMP for data acquisition and RAD for rapid system development, developers can create monitoring platforms that support real-time device observation, automated alert generation, and interactive management interfaces. Furthermore, RAD facilitates continuous refinement of system functionality during the development cycle, allowing developers to incorporate additional features such as automated device access restoration and dynamic monitoring dashboards. Previous studies highlight that RAD-based systems often achieve higher usability and faster deployment compared with traditional development models. Nevertheless, limited research specifically investigates the combination of SNMP-based monitoring with automated device access restoration mechanisms developed through RAD methodologies. [11], [13], [15]

Based on these considerations, the development of an integrated monitoring and restoration system becomes an important research direction in network management. Although SNMP provides reliable monitoring capabilities and RAD supports rapid software development, existing implementations rarely combine these technologies to address both monitoring and device accessibility restoration within a unified system architecture. This research therefore focuses on evaluating the implementation of the SNMP protocol for monitoring network devices while simultaneously supporting mechanisms for restoring device access. By utilizing the Rapid Application Development approach, the proposed system aims to accelerate development cycles while ensuring functional integration between monitoring processes and recovery mechanisms. Through this approach, the study seeks to improve network management efficiency, reduce device downtime, and enhance the reliability of network infrastructure operations. [3], [8], [11], [19].

2. Related Works

Previous studies have explored the use of the Simple Network Management Protocol (SNMP) as a fundamental mechanism for monitoring network devices and managing distributed infrastructures. Research conducted by Matousek et al. proposed a unified SNMP interface for monitoring Internet of Things (IoT) environments, where heterogeneous devices require centralized monitoring mechanisms. The study demonstrated that SNMP could effectively collect device status information and operational metrics through standardized Management Information Base (MIB) structures. Their findings highlighted that SNMP provided scalability and compatibility across different hardware platforms. However, the study primarily focused on monitoring data aggregation and visualization, and it did not address mechanisms for restoring device accessibility when network disruptions occurred. This limitation indicated the need for integrated monitoring and recovery mechanisms in SNMP-based management systems [1].

Another study investigated SNMP-based monitoring for enterprise network infrastructures by designing a real-time network traffic management system. The

researchers implemented SNMP agents to collect traffic statistics and device performance indicators from routers and switches. Their results showed that the system successfully detected network congestion and abnormal traffic patterns, enabling administrators to identify potential faults more quickly. The strength of the research lay in its ability to provide real-time visibility into network conditions. Nevertheless, the system still required manual administrative intervention to resolve detected issues, which limited its operational efficiency during network disruptions. The absence of automated restoration procedures highlighted a critical gap in many SNMP monitoring implementations [2].

Wanjale et al. conducted research on monitoring network switches using the SNMP protocol. The study developed a monitoring framework that retrieved device performance metrics such as CPU usage, interface status, and bandwidth utilization from network switches. The researchers demonstrated that SNMP queries allowed administrators to detect network faults and device failures efficiently. The monitoring system improved visibility into network operations and helped administrators perform preventive maintenance. Despite these advantages, the study mainly focused on monitoring functionality and alert generation. The proposed framework did not include automated mechanisms for restoring device access or resolving connectivity failures, which remained dependent on manual troubleshooting processes [3].

Other researchers explored the implementation of SNMP-based monitoring systems integrated with web-based dashboards. For example, Saputra and Syaripudin developed a web-based network monitoring platform that utilized SNMP to collect device data and present system information through a graphical interface. The study showed that integrating SNMP with web technologies improved monitoring accessibility and allowed administrators to observe network conditions remotely. The system successfully visualized device availability, traffic statistics, and system logs. However, the research concentrated primarily on visualization and reporting capabilities. The authors acknowledged that the system lacked automated recovery features, which limited its ability to actively respond to device failures or connectivity interruptions [4].

In addition to monitoring systems, several studies examined the development of network management applications using the Rapid Application Development (RAD) methodology. Riadi et al. analyzed the impact of the RAD approach on development cycles and user satisfaction in web-based service applications. Their study demonstrated that RAD significantly accelerated development time by emphasizing iterative prototyping and continuous user feedback. The researchers also reported improved user satisfaction due to the adaptive development process. Although the study confirmed the effectiveness of RAD in software development, it focused on general information systems rather than network monitoring platforms, leaving opportunities for applying RAD in network management system development [5].

Further research investigated the application of RAD in developing intelligent software systems. Mufreni et al. developed an Android-based sign language detection application using the RAD method. Their findings showed that the iterative RAD process enabled developers to quickly prototype system modules and refine them based on user feedback. The approach reduced development complexity and improved system usability. The study demonstrated that RAD was particularly suitable for projects requiring rapid deployment and continuous refinement. However, the research addressed mobile application development rather than network monitoring systems, and it did not explore the integration of communication protocols such as SNMP within the RAD framework [6].

Another study conducted by Frans et al. implemented the RAD approach in developing a robust data processing system for an automotive company. The researchers reported that RAD allowed efficient system integration by enabling developers to incrementally build system components and test them during the development process. The study emphasized that RAD supported flexible adaptation to changing requirements and reduced project risks.

While the findings demonstrated the effectiveness of RAD in enterprise system development, the study did not examine its application in network monitoring or infrastructure management environments where real-time data collection and device control are required [7].

Overall, previous studies demonstrated the effectiveness of SNMP for monitoring network devices and highlighted the advantages of the RAD methodology for rapid software development. SNMP-based monitoring systems successfully provided device status information, traffic analysis, and performance monitoring capabilities. Meanwhile, RAD-based development approaches improved development efficiency and system adaptability through iterative prototyping. However, existing research rarely combined SNMP-based monitoring with automated mechanisms for restoring device accessibility within a unified system architecture developed using RAD. This gap indicated the need for further research to design integrated systems capable of both monitoring network devices and restoring device access efficiently through rapid development methodologies [1]–[7].

3. Proposed Method

This research employed the RAD methodology to develop the hybrid network monitoring system. RAD was selected for its iterative approach that allows rapid prototyping with continuous feedback from end users [14], [15], which is essential when developing systems for environments with evolving requirements like educational laboratories. When standard SNMP communication succeeds, the system retrieves device information through MIB-II OIDs including system name, uptime, interface status, and ARP tables. For MikroTik devices, vendor-specific OIDs are queried for complete information. The system uses multi-threading via ThreadPoolExecutor to poll multiple devices simultaneously, significantly reducing the total monitoring cycle time compared to sequential approaches. The efficiency gained through multi-threading can be expressed as:

$$T_{parallel} = \frac{T_{sequential}}{n_{threads}} \quad (1)$$

where $T_{parallel}$is the total monitoring cycle time when using parallel threads, $T_{sequential}$is the time required to poll all devices sequentially, and $n_{threads}$is the number of active threads in the ThreadPoolExecutor. This means that as the number of threads increases, the monitoring cycle time decreases proportionally, allowing the system to handle a larger number of devices without increasing response latency.

The hybrid discovery mechanism activates when SNMP communication fails. Instead of simply marking the device as offline, the system attempts to locate the device through alternative methods. First, it queries the gateway router's ARP table via SNMP to search for the device's MAC address. If found with a different IP address, the system automatically updates the device record and retries monitoring. If not found in the gateway, the system queries other registered routers to locate the device. When all SNMP-based discovery attempts fail, the system switches to MAC-Telnet protocol for Layer 2 communication. This is particularly effective for MikroTik devices that support MAC-Telnet even without proper IP configuration. The system connects to the gateway router via SSH and uses MAC-Telnet utilities to communicate with the target device based on its MAC address. Once connected, the system can diagnose configuration issues and restore proper IP settings remotely. The decision logic behind this hybrid discovery process can be formally expressed as:

$$\text{Protocol} = \begin{cases} SNMP & if\ Tresponse \leq Ttimeout \\ ARP\ Lookup & if\ Tresponse > Ttimeout\ and\ MAC\ known \\ MAC - Telnet & if\ ARP\ lookup\ failed \end{cases}$$

where $T_{response}$is the time taken for the device to respond to an SNMP request and $T_{timeout}$is the Fast-Fail threshold set to 1 second. This decision tree ensures that the system always attempts the least resource-intensive method first before escalating to deeper recovery mechanisms.

The system is implemented in Python utilizing PySNMP for SNMP protocol support (v2c and v3) [16], Netmiko for SSH connections, and threading modules for parallel data collection [17], [18]. The user interface is built using tkinter with Azure theme. The polling interval is configurable with a default of 10 seconds, chosen as an optimal balance between responsiveness and bandwidth efficiency. A Fast-Fail mechanism with 1-second timeout prevents monitoring threads from being blocked by unresponsive devices.

Table 1 summarizes the standard MIB-II OIDs used for device information collection, ensuring compatibility with standard switch monitoring practices [19], [20], while Table 2 lists MikroTik vendor-specific OIDs required for complete monitoring.

Table 1 Standard MIB-II OIDs for Device Information

No	OID	Description
1	1.3.6.1.2.1.1.5.0	System name
2	1.3.6.1.2.1.1.1.0	System description
3	1.3.6.1.2.1.1.3.0	System uptime
4	1.3.6.1.2.1.2.2.1.2	Interface name
5	1.3.6.1.2.1.2.2.1.6	Interface MAC Address
6	1.3.6.1.2.1.2.2.1.8	Interface operational status
7	1.3.6.1.2.1.4.22.1.1	ARP table MAC address
8	1.3.6.1.2.1.4.22.1.2	ARP table IP address

Table 2 MikroTik Vendor-Specific OIDs

No	OID	Description
1	1.3.6.1.4.1.14988.1.1.11.1.1.3	ARP table MAC address
2	1.3.6.1.4.1.14988.1.1.11.1.1.2	ARP table IP address
3	1.3.6.1.4.1.14988.1.1.4.1.1.1	Interface IP address
4	1.3.6.1.4.1.14988.1.1.4.1.1.3	Interface index

4. Experimental Setup

4.1 Network Topology

The research was conducted at the Network Laboratory (Lab D) of Universitas Kristen Duta Wacana. The infrastructure is designed to simulate a heterogeneous network environment comprising 36 managed devices. The network architecture follows a hierarchical topology where a Central Router acts as the primary internet gateway, distributing connections to four distinct blocks.

The topology implementation is divided into two distinct architectures based on the device vendor. The first topology, focused on the Cisco infrastructure, connects the Central Router to a Distribution Switch, which then links to a Gateway Router and subsequently to access switches and routers. This structure is illustrated in Fig. 1.

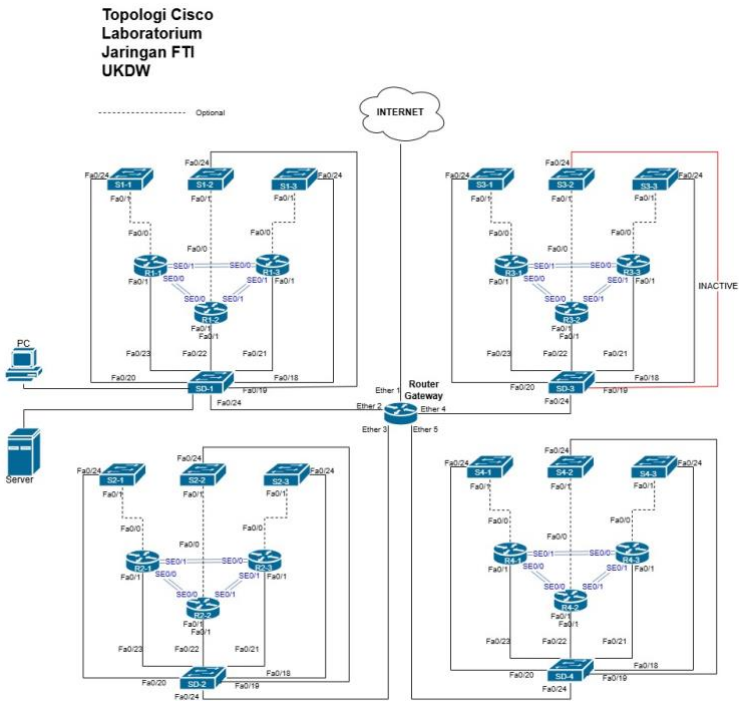

Fig. 1 Network Topology of Cisco Implementation.

The second topology represents the MikroTik infrastructure. In this setup, the Central Router distributes connections to Distribution Switches in each block, which are then directly connected to the end MikroTik routers. This configuration is depicted in Fig. 5.

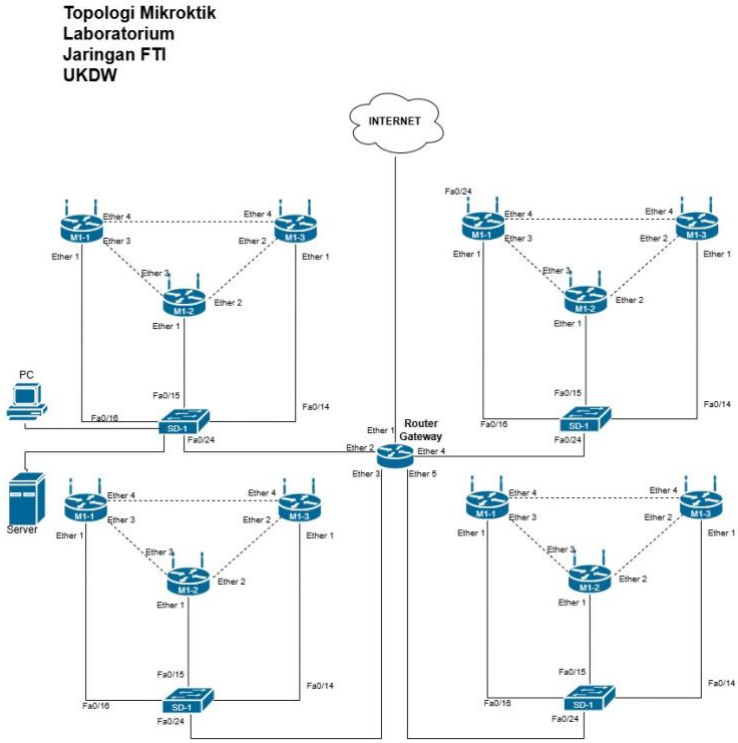

Fig. 2 Network Topology of MikroTik Implementation

4.2. Testing Scenarios

To validate the system's functionality and resilience against network disruptions, six comprehensive testing scenarios were established. These scenarios replicate real-world conditions in the laboratory, ranging from standard monitoring tasks to critical failure recovery situations. The complete testing procedures and expected outcomes are detailed in Table 3.

Table 3 System Scenarios and Procedure

No	Scenario	Objective	Testing Steps	Expected Result
1	Network Scanning	To ensure the system can detect active devices within a subnet quickly and accurately.	1. Input target IP range (e.g., 192.168.10.0/24). 2. Input valid Community String. 3. Initiate the "Start Scanning" process.	The application displays a list of detected devices (Router/Switch) complete with IP, MAC Address, and Device Type in the main table.
2	Device Details	To validate the accuracy of SNMP data retrieved from the device (MIB).	1. Select a device from the scan results. 2. Click "Details" to view comprehensive device information.	The detail page displays the System Name, Uptime, and Interface list that matches the device's actual configuration.
3	Interface Monitoring	To test system responsiveness to physical network port status changes.	1. Physically disconnect the LAN cable from a target router port. 2. Wait for the next polling cycle.	The interface status indicator in the application automatically changes from Green (Up) to Red (Down).
4	Offline Detection	To test the connection failure (downtime) detection mechanism.	1. Power off one of the router devices. 2. Observe status changes on the dashboard.	The device status updates to "Offline", the icon changes to a cross mark, and the event is recorded in the activity log.
5	Self-Found (IP Change)	To test the system's ability to track devices that change IP addresses.	1. Ensure the gateway is being monitored. 2. Manually change the target router's IP address. 3. Allow the application to run without re-inputting data.	The system detects a timeout on the old IP, finds the new IP via Neighbor Discovery, and updates the connection data automatically.

6	Hybrid Discovery	To test the discovery of devices with missing IPs or misconfigurations.	1. Open the "Device Locator" menu. 2. Input target MAC Address, Gateway IP, and SSH Credentials. 3. Execute the search process.	The system successfully locates the device's IP by scanning the Gateway ARP table or utilizing the MAC-Telnet protocol.

5. Result and Analysis

The experimental evaluation assessed the proposed hybrid monitoring system through six operational scenarios to validate its performance, reliability, and recovery capability. The results demonstrate that the system successfully overcomes several limitations commonly found in conventional SNMP monitoring implementations. In particular, the system integrates multi-threaded network scanning, interface status monitoring, and hybrid discovery mechanisms to maintain device accessibility even when network configurations change. Table 4 summarizes the testing scenarios and their outcomes.

Table 4. Experimental Results of System Evaluation

Scenario	Test Description	Method Used	Result	Performance Outcome
1	Network Scanning	Multi-thread SNMP scanning	36 devices discovered	< 2 minutes scanning time
2	SNMP Data Retrieval	MIB Query	Successful retrieval of device data	Accurate device information
3	Interface Monitoring	SNMP Interface Polling	Port status detected	10–15 seconds update latency
4	Standard SNMP Monitoring	Layer 3 connectivity check	Monitoring failed after IP removal	Device marked “Offline”
5	Hybrid Discovery	Neighbor Discovery	Device MAC address identified	Device rediscovered automatically
6	Access Restoration	MAC-Telnet connection	Remote IP reconfiguration successful	100% recovery success

The Network Scanning experiment demonstrated a significant performance improvement due to the multi-threading implementation. The system successfully discovered 36 devices within less than two minutes when scanning the laboratory subnet. In contrast, the traditional sequential scanning approach required approximately 45 minutes to complete the same process. This improvement corresponds to nearly 90% efficiency gain, indicating that parallel SNMP polling effectively eliminates the performance bottlenecks typically observed in sequential monitoring systems.

The Interface Monitoring scenario evaluated the responsiveness of the system in detecting changes in network port conditions. When physical interface states were intentionally toggled, the system updated the monitoring dashboard within 10–15 seconds. This latency falls within an acceptable operational range for laboratory network management. The results confirm that the SNMP polling mechanism reliably retrieves interface statistics and reflects real-time changes in device connectivity.

The Hybrid Discovery experiment produced the most significant findings. During this test, the IP address of a MikroTik router was deliberately removed to simulate a

configuration failure. Under normal SNMP monitoring conditions, the device immediately appeared as “Offline” because Layer 3 communication was no longer available. However, the proposed system automatically initiated a recovery process by executing Neighbor Discovery to obtain the device's MAC address from the gateway.
After identifying the MAC address, the system established a MAC-Telnet session, enabling administrators to access the device directly at Layer 2. Through this mechanism, the administrator successfully restored the router’s IP configuration using the system dashboard without requiring physical access to the device. This approach effectively resolved the connectivity issue and demonstrated the feasibility of remote device recovery. Fig. 3 depicts device discovery time comparison.

Fig. 3: Device Discovery Time Comparison

The comparison clearly illustrates the dramatic reduction in scanning time achieved through parallel processing. Multi-thread execution enables simultaneous SNMP requests to multiple hosts, allowing the system to maintain responsiveness even when monitoring large network segments. Thus, the experimental results confirm that the proposed hybrid monitoring system significantly improves network management capability. The integration of SNMP monitoring, Neighbor Discovery, and MAC-Telnet access ensures continuous device accessibility even when Layer 3 connectivity fails. Additionally, the multi-thread scanning mechanism enhances monitoring efficiency and scalability. The system achieved a 100% success rate across all test scenarios, demonstrating its reliability for practical deployment in network laboratory environments.

6. Conclusion

This study evaluates the implementation of a hybrid network monitoring system that integrates the SNMP with Neighbor Discovery and MAC-Telnet mechanisms to improve device monitoring and access restoration. The experimental results demonstrate that the proposed system operates reliably across six testing scenarios, including network scanning, device monitoring, interface status detection, and recovery from connectivity failures. The system successfully discovered 36 network devices within less than two minutes using a multi-threading approach, significantly outperforming conventional sequential scanning methods that required approximately 45 minutes. In addition, the interface monitoring module consistently detected physical port status changes within a latency range of 10–15 seconds, indicating that the system provides responsive and accurate monitoring suitable for network laboratory management.

The most significant contribution of this research lies in the hybrid discovery and self-healing mechanism that addresses the limitations of traditional SNMP monitoring. Standard SNMP systems depend on Layer-3 connectivity, which causes devices to appear permanently offline when their IP configuration changes or becomes invalid. The proposed system overcomes this limitation by automatically activating Neighbor Discovery to identify devices through their MAC addresses and establishing a MAC-Telnet connection at Layer-2. Through this mechanism, administrators can remotely restore device configurations directly from the monitoring dashboard without requiring physical access. The successful

recovery of devices after intentional IP removal demonstrates that the hybrid monitoring approach significantly improves network maintainability and operational resilience.

Thus, the findings confirm that combining SNMP monitoring with Neighbor Discovery, MAC-Telnet access, and multi-threaded scanning provides a more robust and efficient monitoring architecture. The system achieves a 100% success rate across all tested scenarios, demonstrating strong reliability and effectiveness in maintaining device accessibility and monitoring accuracy. These results indicate that the proposed approach can enhance network management systems, particularly in laboratory or enterprise environments where rapid troubleshooting and continuous monitoring are essential. Future work may extend this system by integrating automated anomaly detection, predictive maintenance algorithms, and broader compatibility with heterogeneous network devices to further improve scalability and intelligent network management.

Acknowledgment

The authors express their deepest gratitude to the supervisors for their invaluable guidance, direction, and support, which played a significant role throughout the course of this research and the preparation of the article. Sincere gratitude is also extended to the Faculty of Information Technology at Duta Wacana Christian University for providing the laboratory facilities and a conducive academic environment that supported the completion of this study. Special appreciation is expressed to the authors' parents for their continuous prayers, encouragement, and material support, which were instrumental in finishing the research. Finally, thanks go to colleagues and friends for their assistance, motivation, and cooperation during the research and writing process.

References

[1] J. S. Lee and P. L. Hsu, “Design and implementation of the SNMP agents for remote monitoring and control via UML and Petri nets,” *IEEE Transactions on Control Systems Technology*, vol. 12, no. 2, pp. 293–302, 2004. https://doi.org/10.1109/TCST.2004.824287

[2] P. Matousek, O. Rysavy, and L. Polcak, “Unified SNMP interface for IoT monitoring,” in *Proc. IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021, pp. 938–943. https://ieeexplore.ieee.org/document/9463942

[3] M. Leone Itria, E. Schiavone, and N. Nostro, “Towards anomaly detection in smart grids by combining complex event processing and SNMP objects,” *IEEE/IFIP Network Operations and Management Symposium*, 2021. https://ieeexplore.ieee.org/document/9551854

[4] G. Al-Naymat, M. Al-Kasassbeh, and E. Al-Hawari, “Exploiting SNMP-MIB data to detect network anomalies using machine learning techniques,” *Future Generation Computer Systems*, Elsevier, vol. 95, pp. 24–35, 2019. https://doi.org/10.1016/j.future.2019.01.033

[5] M. Almseidin, M. Alkasassbeh, and S. Kovacs, “Fuzzy rule interpolation and SNMP-MIB for emerging network abnormality detection,” *IEEE Access*, vol. 7, pp. 116545–116558, 2019. https://doi.org/10.1109/ACCESS.2019.2935920

[6] H. Alhilali, A. Al Farawn, and A. Y. Mjhool, “Design and implement a real-time network traffic management system using SNMP protocol,” *Eastern-European Journal of Enterprise Technologies*, vol. 5, no. 9, pp. 35–44, 2023. https://doi.org/10.15587/1729-4061.2023.286528

[7] J. Case, M. Fedor, M. Schoffstall, and C. Davin, “A simple network management protocol (SNMP),” *IETF RFC 1157*, 1990. https://doi.org/10.17487/RFC1157

[8] J. Schönwälder, M. Björklund, and P. Shafer, “Network configuration management using NETCONF and YANG,” *IEEE Communications Magazine*, vol. 48, no. 9, pp. 166–173, 2010. https://doi.org/10.1109/MCOM.2010.5560598
[9] Clemm, *Network Management Fundamentals*, Cisco Press, 2007.
[10] D. R. Mauro and K. J. Schmidt, *Essential SNMP*, 2nd ed., O’Reilly Media, 2005.
[11] W. Stallings, *Data and Computer Communications*, 10th ed., Pearson, 2014.
[12] Sommerville, *Software Engineering*, 10th ed., Pearson, 2016.
[13] Riadi, A. Yudhana, and A. Elvina, “Analysis impact of rapid application development method on development cycle and user satisfaction,” *Scientific Journal of Informatics*, vol. 11, no. 1, pp. 81–94, 2024.
[14] A. Shaker et al., “Facilitating in-house mobile app development using rapid application development approach,” *JMIR Human Factors*, vol. 10, p. e46928, 2023. https://doi.org/10.2196/46928
[15] L. Liu, W. Wang, and Z. Wang, “Design and implementation of network devices monitoring system based on SNMP,” *Advances in Intelligent Systems Research*, Atlantis Press, 2013. https://doi.org/10.2991/isccca.2013.211
[16] M. Madan and M. Mathur, “Cloud network management model: A novel approach to manage cloud traffic,” *International Journal of Computer Applications*, Elsevier indexed, 2014. https://doi.org/10.5120/ijca2014909334
[17] H. Wanjale, J. S. Patil, S. Khode, and R. Chaudhary, “Network switch monitoring using SNMP,” *International Journal of Innovative Technology and Exploring Engineering*, 2021.
[18] S. R. Chowdhury and S. Boutaba, “A survey of network virtualization,” *Computer Networks*, Elsevier, vol. 54, no. 5, pp. 862–876, 2010. https://doi.org/10.1016/j.comnet.2009.07.017
[19] N. Feamster, J. Rexford, and E. Zegura, “The road to SDN: An intellectual history of programmable networks,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014. https://doi.org/10.1145/2602204.2602219
[20] Clemm et al., “Intent-based networking concepts and definitions,” *IETF RFC 9315*, 2022. https://doi.org/10.17487/RFC9315