International Journal of Inventive Engineering and Sciences (IJIES)
ISSN: 2319-9598 (Online), Volume-13 Issue-4, April 2026

Platform-Independent Performance Validation of NS-2.35: A Comparative Study of Native and Virtualized Environments

Vinay Sahu, Ankur Khare, Rani Sahu

Abstract: ***The paper presents a comprehensive experimental study that evaluates the Network Simulator 2 (NS-2.35) on both native Linux systems and VMware-based virtualised environments. Researchers use NS-2.35 as a standard tool for testing Mobile Ad Hoc Networks (MANETs). The researchers face challenges because they must install their software to achieve functionality across multiple operating systems. The research study provides a complete installation guide and troubleshooting instructions, and tests whether different execution environments affect simulation results. The researchers create a MANET scenario using the Ad hoc On-Demand Distance Vector (AODV) routing protocol to conduct identical tests in both environments. The assessment uses Throughput, End-to-End Delay, Packet Delivery Ratio (PDR), and Packet Loss Ratio (PLR) as performance indicators to evaluate node densities of 20, 30, and 50. The researchers conduct multiple experiments and analyse them using the mean and standard deviation (Mean ± SD) to obtain statistically reliable results. The testing results show that all performance measurements remain the same across both native systems and virtualised environments, proving that NS-2.35 delivers platform-independent performance that remains constant under the same system conditions. The test runs produce stable simulation results because two tests show extremely low standard deviation values. The simulation results remain accurate because execution efficiency and system performance differences between the two systems result in only minor discrepancies in the test results. The study concludes that researchers can use both native Linux and virtualised environments for NS-2.35 simulations, as virtualisation preserves simulation fidelity. This work validates the cross-platform performance of NS-2.35, thereby building trust in simulation research findings.***

Keywords: Network Simulator 2 (NS-2.35), Mobile Ad Hoc Networks (MANETs), Virtualization, Native Linux Environment, AODV Routing Protocol, Performance Evaluation, Platform Independence

Nomenclature:
AODV: Ad hoc On-Demand Distance Vector
PDR: Packet Delivery Ratio
VM: Virtual Machine
CBR: Constant Bit Rate
UDP: User Datagram Protocol
PLR: Packet Loss Ratio

Manuscript received on 04 March 2026 | First Revised Manuscript received on 20 March 2026 | Second Revised Manuscript received on 05 April 2026 | Manuscript Accepted on 15 April 2026 | Manuscript published on 30 April 2026.

*Correspondence Author(s)
Vinay Sahu, Research Scholar, Department of Computer Science and Information Technology, Rabindranath Tagore University, Raisen, (Madhya Pradesh), India. Email ID: sahu.vinay@gmail.com, ORCID ID: 0009-0001-9551-4295
Dr. Ankur Khare, Assistant Professor, Department of Computer Science and Information Technology, Rabindranath Tagore University, Raisen, (Madhya Pradesh), India. ORCID ID: 0000-0002-3675-9735
Dr. Rani Sahu*, Associate Professor in Computer Engineering & Applications at IES College of Technology, Bhopal, (Madhya Pradesh), India. Email ID: rani.princy28@gmail.com, ORCID ID: 0000-0001-7844-2085

I. INTRODUCTION

Mobile Ad Hoc Networks (MANETs) have become an important area of research in wireless communication due to their decentralized nature, dynamic topology, and lack of fixed infrastructure. These characteristics make MANETs suitable for applications such as military communication, disaster recovery, and connectivity in remote areas. The design and evaluation of routing protocols in these networks are challenging because the networks' operational patterns continuously shift. Therefore, reliable and flexible simulation tools are required to study their performance effectively [1]. The Network Simulator 2 (NS-2.35) software functions as a popular network protocol modelling and analysis tool that researchers use to study Mobile Ad Hoc Network (MANET) scenarios [2]. The software provides an open-source platform that enables academic and research institutions to study various network protocols, as it supports multiple protocols. Using NS-2.35, the researchers can study critical performance indicators, which include throughput, end-to-end delay, packet delivery ratio, and packet loss, to observe different network behaviour patterns [3].

The installation and configuration processes for NS-2.35 pose significant challenges for users despite the software's benefits. The simulator relies on older libraries and software components that may not work correctly with current operating system designs. Users experience multiple problems, including compiler mismatches, Tcl/Tk version conflicts, and environment configuration errors. Researchers prefer virtualization tools such as VMware Workstation to establish controlled Linux environments for their experiments on systems that lack native Linux support [4]. The process of virtualisation enables easier system deployment, thereby allowing greater operational flexibility, yet there remains a fundamental question about whether this technology affects the accuracy of simulation outcomes. Existing research studies focus on examining installation techniques or evaluating system performance using fundamental testing methods, which assume that native systems will outperform other technology solutions. Research on this topic remains limited because researchers have not investigated the impact of execution platforms on simulation results when operating under identical system configurations.

This study investigates the performance of NS-2.35 by testing it on native Linux

systems and in virtualised environments. The researchers established a MANET environment for testing purposes, using the Ad hoc On-Demand Distance Vector (AODV) routing protocol under standardised testing conditions. The assessment process evaluates performance against primary performance indicators, including throughput, end-to-end delay, packet delivery ratio, and packet loss ratio, across various node-density levels.

The research team conducted multiple simulation experiments to collect data, which they assessed using statistical techniques, including mean and standard deviation calculations. The main objective of this work is not only to compare performance but also to verify whether the simulation results remain consistent across different platforms [5].

The research finding provides valuable information to researchers because it demonstrates that simulation results remain consistent across simulation execution environments. The current study establishes platform-independent validation for NS-2.35 by testing identical MANET scenarios in both native Linux and virtualised environments. The study shows that simulation results remain unchanged across different execution platforms, proving that NS-2.35 delivers consistent, reproducible results across all system architectures.

II. LITERATURE REVIEW

Network simulation reliability is an important research area that directly affects the accuracy of results in both native and virtualized environments. Virtualisation has become widely used for its flexibility and scalability, enabling researchers to test complex network configurations efficiently. Popular simulators such as NS-2, NS-3, and OMNeT++ support detailed wireless network analysis; however, virtualization introduces minor overheads such as CPU scheduling delays and memory management latency. The discrete-event simulator NS-2.35 operates independently of real-time system clocks, ensuring consistent and repeatable results across different platforms. NS-2.35 is widely used in academic research to evaluate MANET routing protocols due to its reliability for packet-level simulation of wireless networks [6]. Virtualisation platforms such as VMware and VirtualBox provide reproducible testing environments, but introduce slight performance overheads compared to native systems [7]. Despite this, modern virtual machines can achieve near-native performance for computational workloads. Among routing protocols, AODV has been extensively studied due to its reactive nature and adaptability in MANET environments. Simulation studies using NS-2 and NS-3 indicate that performance metrics such as Packet Delivery Ratio (PDR), throughput, and delay are significantly influenced by node mobility, density, and traffic patterns. However, most existing studies assume uniform testing environments and do not evaluate the impact of platform differences on simulation outcomes. Recent research highlights reproducibility challenges in network simulations. Although NS-2.35 is theoretically platform-independent, virtualisation introduces subtle variations in CPU scheduling, process handling, and memory allocation, leading to differences in simulation execution times [8]. These variations can affect time-dependent performance metrics and overall consistency of results. Therefore, there is a need for a systematic evaluation of NS-2.35 across native Linux and virtualised environments to ensure reliable and reproducible results. Existing studies have mostly treated routing protocol performance and simulation platform analysis separately, with limited focus on their combined impact. This gap motivates the present study [9].

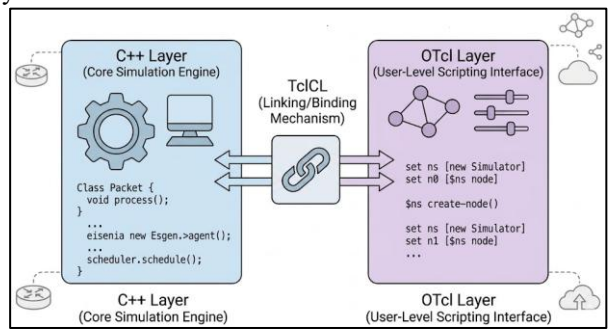
III. BACKGROUND

The NS-2.35 simulation tool is a discrete event-driven simulator that engineers use to build models and conduct research in wireless networks and Mobile Ad Hoc Networks (MANETs). Researchers at academic institutions, together with those who evaluate routing algorithm performance, have selected NS-2.35 as their primary testing platform because it provides flexible design options and supports multiple network protocols. The current simulation environment uses NS-2.35 because it offers complete packet-level testing, detailed documentation, and a vast collection of simulation scripts. The system includes features that enable it to conduct controlled research studies that need both reproducibility and complete research assessment.

A. NS-2.35 Architecture

The NS-2.35 system uses a dual-language programming architecture that combines C++ performance capabilities with OTcl (Object Tool Command Language) programming flexibility [10]. The system architecture of the combined system enables users to perform their simulation experiments with two advantages: high-performance processing and simple setup procedures.

The main simulation engine is implemented in C++, performing essential functions such as protocol implementation, event scheduling, packet processing, and memory resource management. This strategy enables the system to run extensive simulations while accurately simulating network patterns. Figure 1 illustrates the dual-layer architecture of NS-2.



[Fig.1: NS2 Dual-Layer NS-2 Architecture [11]]

The OTcl scripting interface enables users to create network topologies, set simulation parameters, and manage simulation execution without modifying C++ code. The system design enables users to configure simulations because simulation settings exist separately from the main system functions.

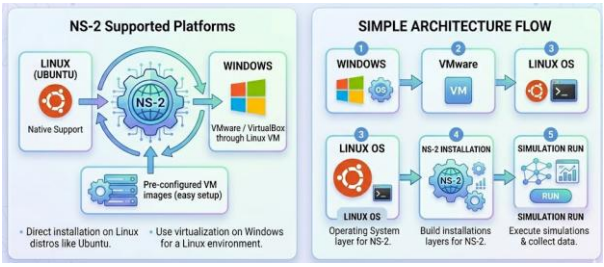
TclCL acts as a linking mechanism between C++ and OTcl, enabling communication between

compiled components and the scripting interface. The NS-2.35 system uses a dual-layer architecture that provides both efficient execution and flexible configuration to support network research and simulation studies.

B. Supported Platforms

NS-2.35 was originally designed for Unix operating systems and performs best on Linux Ubuntu distributions, which provide stable, dependable performance. The system cannot operate on Windows platforms because it requires Unix-based system libraries and system calls to function properly. Non-Linux systems can create Linux-based environments using virtualisation tools such as VMware Workstation and VirtualBox. This allows NS-2.35 to be executed even on systems where direct installation is not possible [12]. Organisations increasingly depend on pre-configured virtual machines and container-based environments to streamline deployment, reducing installation time. Users can access these methods, but they introduce another layer of separation between the simulator and the physical system.

Figure 2 illustrates the supported platforms for NS-2 along with its basic installation workflow. The system demonstrates direct operational capability on Linux (Ubuntu) systems. At the same time, Windows users can access it through virtualisation software such as VMware and VirtualBox, enabling them to complete the entire process from installation to simulation execution.



[Fig.2: NS-2 Supported Platforms and Architecture Flow [13]]

Native Linux environments offer better performance by enabling users to access hardware directly while consuming minimal system resources. The study investigates how different platforms affect simulation results, as platform selection requires a comprehensive examination.

C. Limitations

The network simulation research community extensively uses NS-2.35. The network simulation research community extensively uses NS-2.35, yet the system faces multiple constraints due to its reliance on an outdated design and obsolete software components. Modern operating systems face compatibility issues because the simulator requires users to install outdated versions of GCC and Tcl/Tk. Users encounter difficulties with installation and configuration tasks because the system does not function properly with the current platform updates. The installation process requires significant time and effort because it involves multiple tasks, including managing dependencies, building the system, and creating the operational environment. New users need advanced technical skills to complete these tasks because they require specialized knowledge. The NS-2.35 system supports traditional networking models but lacks built-in support for modern technologies such as software-defined networking and large-scale, heterogeneous network systems. The system development process has slowed, resulting in limited capacity for implementing new features. The system requires both C++ and OTcl scripting, making it harder to learn and increasing the complexity of system operations. A complete installation of NS-2.35 with the correct configuration settings is the primary requirement for achieving stable, reliable simulation results.

D. System Requirements

The successful operation of NS-2.35 requires both correct hardware setup and proper software installation. The installation process requires specific system configurations because the system relies on outdated development tools and legacy libraries to function properly and produce accurate simulation results.

The study uses NS-2.35, the standard stable version, to evaluate MANET networks in the simulator. The system requirements are categorised into two environments: a native Linux installation and a virtual machine-based setup.

E. Requirements for Native Linux Installation

For direct installation on a Linux-based system, a compatible and stable distribution is required. Ubuntu (16.04/18.04/20.04 LTS) is generally preferred due to better support for legacy dependencies [14].

F. Hardware Requirements

i. Processor: Dual-core or higher (≥ 2.0 GHz recommended)
ii. RAM: Minimum 2 GB (4 GB or higher recommended)
iii. Storage: At least 10–20 GB of free disk space

G. Software Requirements

i. Operating System: Ubuntu Linux (LTS versions preferred)
ii. Compiler: GCC/G++ (preferably compatible with older versions, e.g., 4.8.x)
iii. Libraries and Tools:
- Tcl/Tk libraries
- Xgraph and NAM visualization tools
- Build-essential packages (make, autoconf, automake, etc.)

Native installation provides direct access to system hardware, ensuring efficient execution, reduced latency, and better control over system resources.

H. Requirements for Virtual Machine (VM) Setup

For users on non-Linux platforms, virtualisation offers a practical and flexible way to run NS-2.35. In this approach, a complete Linux environment is created within the host system using virtualization software such as VMware Workstation [15].

I. Virtualization Requirements

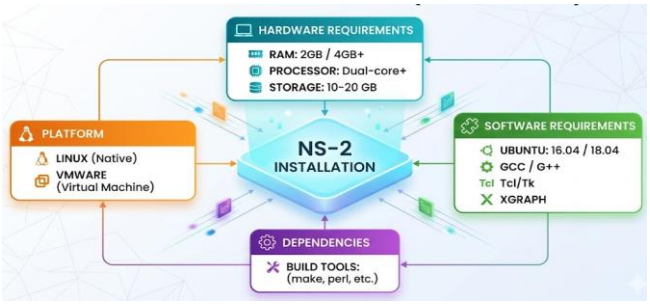
i. Virtualization Software: VMware Workstation / VirtualBox (or equivalent)
ii. OS Image: Ubuntu ISO (recommended 16.04 or 18.04)

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

J. Virtual Machine Configuration

i. *RAM Allocation:* Minimum 2 GB (4 GB recommended)
ii. *Storage Allocation:* At least 20 GB virtual disk space
iii. *Processor Allocation:* Minimum 2 CPU cores

Virtual machines require their resources to be allocated properly because this process establishes the necessary conditions for stable performance and execution speed. Virtualisation introduces additional overhead through its system resource abstraction, yet this implementation improves the user experience, enhances system portability, and simplifies installation for new users who do not use Linux. Figure 3 shows the main requirements for installing NS-2, including hardware, software, platform options (Linux or VMware), and essential dependencies.



[Fig.3: NS-2 Installation Requirements and Prerequisites [16]]

IV. INSTALLATION METHODOLOGY

The section describes a complete, repeatable method for installing and setting up NS-2.35 on a Linux system. The methodology creates standardised installation procedures to minimise installation errors caused by outdated dependencies and system compatibility issues. The study uses the NS-2.35.35 all-in-one package for installation.

A. Native Linux Installation

The process of installing NS-2.35 on a native Linux system begins with system preparation and ends with installing the required dependencies, compiling, and configuring the system. The installation process must be executed correctly, with all steps completed properly, to ensure a successful installation and maintain reliable simulator operation. The NS-2 installation procedure starts with system setup and continues through dependency installation, package building, configuration, and execution, as shown in Figure 4 [17].

[Fig.4: Step-by-Step NS-2 Installation Workflow (Author's own work)]

B. System Update

Before installation, the system repositories and existing packages are updated to ensure compatibility with required dependencies. This step helps prevent conflicts caused by outdated or missing packages and prepares the system for subsequent installations.

C. Dependency Installation

NS-2.35 relies on several legacy libraries and development tools. Essential dependencies, including compilers (GCC/G++), build utilities, Tcl/Tk libraries, and visualization tools such as Xgraph and NAM, are installed to support compilation and execution. Proper dependency management is crucial to avoiding installation failures and runtime errors.

D. NS-2.35 Package Extraction

The NS-2.35.35 all-in-one package is downloaded from a reliable source and extracted into the working directory. This package contains all necessary components, including the simulation engine, libraries, and supporting tools required for installation.

E. Compilation and Installation

The installation is initiated by executing the provided installation script, which compiles the NS-2.35 source code along with its associated libraries. This process may generate warnings due to compatibility issues with modern systems; however, successful completion ensures that the simulator binaries are properly built and configured.

F. Environment Configuration

After installation, environment variables such as the system PATH and library paths are configured to enable seamless execution of NS-2.35 commands. Proper configuration ensures that the system can locate the simulator binaries and associated libraries without manual intervention.

G. Installation Verification

The installation is verified by executing the NS-2.35 command-line interface. Successful invocation of the simulator prompt confirms that the installation and configuration processes have been completed correctly. Additionally, basic simulation scripts may be executed to validate the functional integrity of the setup.

H. VMware-Based Installation

The installation process for NS-2.35 on Windows systems needs VMware Workstation virtualization to create its required virtual environment [18]. The approach establishes a controlled Linux environment, enabling the simulator to operate without compatibility issues arising from the host system. The complete installation process of NS-2 on a Windows system using VMware is shown in Figure 5. The process consists of creating a virtual machine, installing necessary software dependencies, extracting packages, configuring the environment, and performing system validation.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

[Fig.5: NS-2 Installation via VMware and Ubuntu Workflow (Author's Own Work)]

I. Virtual Machine Creation

The virtualization platform enables users to create a virtual machine (VM) by selecting standard configuration settings and loading an appropriate Linux distribution image. This step establishes the foundational environment required for NS-2.35 installation.

J. Resource Allocation

The virtual machine receives an appropriate allocation of system resources, including CPU cores, memory, and storage, to ensure stable performance. Fixed resource allocation is preferred to minimize runtime variability and maintain consistency during simulation execution.

K. Ubuntu Installation

The virtual machine contains an installed Linux operating system, typically Ubuntu. The installation process includes system setup, user configuration, and initialization of essential system services required for further software deployment.

L. Dependency Setup

The virtual environment is configured to install all necessary development tools and libraries required by NS-2.35 after the operating system installation. The installation process includes compilers, Tcl/Tk libraries, and supporting utilities that enable users to work with the NS-2.35 framework.

M. NS-2.35 Installation

The NS-2.35.35 all-in-one package is deployed in the virtual machine, and the installation is performed using standard compilation procedures. The simulator and its components are properly installed when all tasks reach a successful status in the virtualised environment.

N. Configuration and Verification

NS-2.35 commands execute seamlessly because environment variables have been configured. The installation is verified by launching the simulator interface and executing basic simulation scripts to confirm correct functionality and system stability.

V. COMMON ISSUES AND SOLUTIONS

The installation and configuration of NS-2.35 often present several challenges due to its reliance on legacy software components and outdated dependencies. These issues are particularly prominent when deploying NS-2.35 on modern operating systems. This section outlines the most common problems encountered during installation, along with their causes and effective solutions.

A. Tcl/Tk Compatibility Issues

One of the most frequently encountered issues is incompatibility between NS-2.35 and modern versions of the Tcl/Tk libraries.

i. Cause:

NS-2.35 depends on older versions of Tcl/Tk, whereas contemporary Linux distributions include updated versions that may not be fully compatible.

ii. *Solution:*

Installing compatible or downgraded versions of Tcl/Tk or modifying configuration settings to align with the system libraries effectively resolves this issue.

B. GCC Compiler Issues

Compilation errors often arise from incompatibility between the NS-2.35 source code and modern compiler versions.

i. Cause:

Recent GCC versions may not support certain legacy constructs used in NS-2.35.

ii. Solution:

Using an earlier compatible compiler version (e.g., GCC 4.8. x) during the build process ensures successful compilation.

C. Xgraph and NAM Errors

Visualization tools such as Xgraph and NAM may fail to execute properly after installation.

i. Cause:

Missing graphical libraries or incomplete installation of required dependencies.

ii. Solution:

Installing necessary graphical libraries (e.g., X11-related packages) and ensuring proper display configuration resolves these issues.

D. Permission Issues

Permission-related errors may occur during installation or execution.

i. Cause:

Restricted access to system directories or insufficient user privileges.

ii. Solution:

Executing commands with administrative privileges and ensuring appropriate file permissions mitigates such errors.

E. Environment Variable Issues

An improper configuration of environment variables can prevent NS-2.35 from executing even after a successful installation.

i. Cause:

Incorrect or missing PATH and library path definitions.

ii. Solution:

Properly defining environment variables in system configuration files and reloading them ensures correct execution.

F. Installation Failures

In some cases, the installation process may terminate prematurely or fail during compilation.

i. Cause:

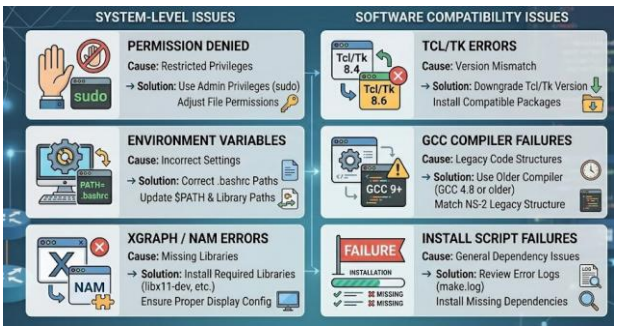
Missing dependencies, incompatible libraries, or incorrect system configuration.

ii. Solution:

Careful examination of installation logs, identification of missing components, and iterative correction of dependencies typically resolve the issue.

G. Summary of Issues and Troubleshooting

The installation of NS-2.35 is difficult because the system relies on outdated software components, and its design predates previous technologies. Successful setup typically requires proper dependency handling, a compatible toolchain, and careful system configuration to resolve common issues. Establishing a structured troubleshooting system enables organisations to improve their installation process and increase their deployment success rates. The findings of this study demonstrate that researchers need to maintain stable experimental settings to conduct reliable simulation studies [19]. Figure 6 shows a troubleshooting guide that describes common installation problems with NS-2, along with their solutions, including both system-level and compatibility-related issues [20].



[Fig.6: NS-2 Common Installation Issues and Solutions]

VI. PERFORMANCE EVALUATION AND RESULTS

A. Simulation Environment and Experimental Setup

The study uses NS-2.35 (version 2.35) to evaluate performance from both native Linux systems and virtualized environments. The two testing environments use the same simulation settings to achieve unbiased evaluation results. The testbed uses the AODV routing protocol to study Mobile Ad Hoc Network (MANET) operations under predetermined testing conditions. The simulation scenarios use different node densities to assess how network performance varies across different traffic conditions. The researchers run each configuration multiple times to obtain reliable statistical results that are uniform across different experiment sessions.

VII. SIMULATION PARAMETERS

The experimental tests use standard parameters in their simulations, which require keeping testing conditions identical throughout all test sessions. The study requires specific parameters, including node density, simulation area, traffic characteristics, and protocol configurations, to be selected. The evaluation process uses these settings to create consistent testing conditions. The complete list of simulation parameters used in this work is provided in Table I.

Table I: Simulation Parameters (Source: Author's Experimental setup)

Parameter	Value
Simulator	NS-2.35
Network Type	MANET
Routing Protocol	AODV
Number of Nodes	20, 30, 50
Simulation Area	500 m × 500 m
Simulation Time	100 seconds
Mobility Model	Random Waypoint
Node Speed	2–10 m/s
Traffic Type	CBR over UDP
Packet Size	512 bytes
Packet Interval	0.1 seconds
MAC Protocol	IEEE 802.11
Propagation Model	Two-Ray Ground

These parameters are selected to reflect realistic MANET conditions while ensuring controlled experimentation.

A. Network Topology and Node Deployment

The network topology evaluation tests both the MANET system's ability to grow and its capacity to maintain stable performance. The simulation creates a realistic network environment by randomly placing nodes throughout a 500 m × 500 m space to study network density. The researchers tested the effects of node density by creating three scenarios that used 20, 30, and 50 nodes, respectively. Nodes move through the network according to the Random Waypoint Model, which allows them to travel at speeds between 2 m/s and 10 m/s. The system accurately demonstrates how MANETs operate by tracking both ongoing changes in network topology and intermittent link failures. The complete set of network topology configuration parameters is summarised in Table II.

Table II: Topology and Mobility Configuration (Source: Author's Experimental Setup)

Parameter	Specification
Topology Type	Wireless Ad Hoc (MANET)
Node Population	20, 30, 50 (Scalability Test)
Deployment Area	500 m × 500 m
Node Placement	Random Distribution
Mobility Model	Random Waypoint
Velocity	2 m/s – 10 m/s
Network Boundary	Fixed (Confined Area)

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

B. Traffic Model (CBR over UDP)

The evaluation of data transmission performance uses a Constant Bit Rate (CBR) traffic model over User Datagram Protocol (UDP) to assess network performance. The researchers selected this combination because it enables them to evaluate routing performance through direct testing while keeping extra protocol costs to a minimum. The simulation uses fixed-size 512-byte packets that are transmitted every 0.1 seconds to create continuous network traffic throughout the testing period. The researchers schedule traffic generation to start and end at specific times, which helps them obtain precise performance metric assessments. Table III shows the complete traffic configuration used by the simulation.

Table III: Traffic Generation Parameters (Source: Author's experimental setup)

Parameter	Specification
Traffic Pattern	Constant Bit Rate (CBR)
Transport Layer	UDP
Source–Destination Pairs	1 Connection
Packet Transmission Rate	10 packets/s
Data Payload Size	512 bytes
Flow Activation Time	2.0 s
Flow Deactivation Time	98.0 s
Data Rate	40.96 kbps (≈ 41 kbps)

C. Routing Protocol Configuration

The Ad hoc On-Demand Distance Vector (AODV) routing protocol operates at the network layer and implements reactive routing techniques. The network establishes routes only when necessary because the dynamic network environment does not require permanent route maintenance, thereby reducing control overhead costs. AODV uses sequence numbers to ensure loop-free routing and to maintain the most recent route information. The system uses Route Request (RREQ) and Route Reply (RREP) messages for route discovery, while link-failure detection mechanisms manage route maintenance. The AODV protocol effectively handles dynamic network topologies and provides a dependable method for testing network performance across various scenarios. Table IV lists all operational parameters used to implement AODV in our study.

Table IV: AODV Protocol Configuration (Source: Author's Experimental Setup)

Parameter	Specification
Protocol Name	Ad hoc On-Demand Distance Vector (AODV)
Routing Strategy	Reactive (On-Demand)
Hello Interval	1.0 seconds
Active Route Timeout	3.0 seconds
Allowed Hello Loss	2 packets
Route Discovery Retries	3
Sequence Numbering	Destination-based (ensures loop freedom)
Route Maintenance Mechanism	Link break detection using HELLO messages
Control Packets	RREQ, RREP, RERR
Route Establishment	On-demand via Route Request (RREQ) and Route Reply (RREP)
Loop Prevention	Sequence numbers and freshness criteria

D. System Configuration and Testbed Setup

The experiments in this study are conducted on two platforms: a native Linux system and a virtualised environment. The purpose of this study is to test whether different execution environments impact simulation results. The two setups use identical parameters, which enables researchers to conduct their experiments under controlled conditions. The controlled testbed design enables researchers to assess NS-2.35 performance across various system architectures using identical input conditions. The main objective is to determine whether the underlying platform affects key simulation metrics, holding all other factors constant.

E. Native Linux Environment

The simulations operate on their main platform, Linux, through the native Linux environment. In this configuration, the NS-2.35 software runs directly on the host operating system, enabling users to access all hardware components, including CPU, memory, and storage. The simulator runs on the physical machine, enabling it to directly control hardware resources because no virtualisation layer would introduce extra processing overhead during simulation activities such as trace file creation and event scheduling. The system achieves efficient performance by operating at lower latency and utilising system resources more effectively. Researchers prefer to use the native environment for simulation studies because it provides stable conditions that yield accurate, high-fidelity results. Table V contains the complete hardware and software configuration details used in the native setup.

Table V: Native Host System Specifications (Source: Author's Experimental Setup)

Component	Specification
Operating System	Ubuntu 20.04 LTS (Focal Fossa)
Kernel Architecture	Linux Kernel 5.x.x (Generic)
Processor	Intel Core i5/i7 (Quad-Core, ≥ 2.5 GHz)
Instruction Set	64-bit (x86_64)
System Memory (RAM)	8 GB DDR4
Storage Media	256 GB Solid State Drive (SSD)
Simulator Version	Network Simulator 2 (NS-2.35)
Toolchain / Compiler	GCC/G++ 4.8.x

F. Virtualized Environment using VMware Workstation

The virtualized setup uses VMware Workstation to create a Linux-based virtual machine on the host system. The system establishes a virtualization layer that protects the NS-2.35 simulator from direct access to the physical system resources. The virtual machine operates with exclusive access to processing units and memory resources, ensuring that the research environment maintains its experimental conditions throughout the study. The setup achieves a more consistent operating environment by managing system resources to decrease the effect of hardware sharing on performance. The resource management and scheduling functions of virtualisation add overhead, but the technology provides benefits for application portability across systems and enhanced security and operational flexibility. The program offers particular value to users who operate on systems that lack built-in Linux support. The virtualized testing environment uses Table VI to describe its complete setup.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Table VI: VMware Guest System Resource Partitioning (Source: Author's Experimental Setup)

Component	Specification
Hypervisor Type	VMware Workstation Pro (Type-2 Hypervisor)
Guest Operating System	Ubuntu 14.04 LTS (Trusty Tahr)
Virtual Processor (vCPU)	2 Cores (Dedicated)
Memory Allocation	4 GB SDRAM (Reserved/Locked)
Disk I/O Architecture	Virtual SCSI (20 GB Provisioned Storage)
Network Interface	Network Address Translation (NAT) Mode
Hardware Acceleration	Intel VT-x Enabled

G. Performance Evaluation Metrics

To quantitatively evaluate the network performance and analyse the impact of execution environments, key performance metrics are considered. These metrics provide insights into data transmission efficiency, network reliability, and latency characteristics of the MANET scenario simulated using NS-2.35.

H. Throughput

Throughput is the total amount of data successfully received at the destination over a given period. It reflects the network's effective data transmission capacity and is typically expressed in bits per second (bps).

$$Throughput = \frac{Total\ Received\ Data\ (bits)}{Simulation\ Time\ (seconds)} \qquad (1)$$

Higher throughput indicates better network performance and efficient utilization of available bandwidth.

I. End-to-End Delay

End-to-End (E2E) Delay is the average time a data packet takes to travel from the source to the destination. It includes transmission, propagation, queuing, and processing delays.

$$D_{avg} = \frac{\sum_{i=1}^{N} t_{receive,i} - t_{send,i}}{N} \qquad (2)$$

where N is the total number of successfully received packets.

Lower delay values indicate faster and more efficient packet delivery.

J. Packet Delivery Ratio (PDR)

Packet Delivery Ratio (PDR) measures network reliability by calculating the ratio of successfully received data packets to the total number of packets sent by source nodes.

$$\text{PDR} = \frac{\text{Total Packets Received}}{\text{Total Packets Sent}} \times 100 \qquad (3)$$

A higher PDR value indicates reliable communication and efficient routing performance.

K. Packet Loss Ratio (PLR)

Packet Loss Ratio (PLR) quantifies the percentage of packets that fail to reach the destination. It is an important indicator of network congestion and instability.

$$\text{Packet Loss} = \frac{\text{Toatl Lost Packets}}{\text{Total Sent Packets}} \times 100 \qquad (4)$$

Lower PLR values indicate better network reliability and minimal data loss.

L. Experimental Results and Analysis

The research experiments were performed using NS-2.35 under identical conditions on both native Linux systems and virtualised computing environments. The researchers conducted multiple simulation tests for each scenario, enabling them to obtain statistically valid results and improve accuracy. Researchers examine how node density and execution environment conditions affect network performance by testing throughput, end-to-end delay, packet delivery ratio, and packet loss ratio. Statistical validation of results is achieved through the calculation of mean and standard deviation, which serve as additional validation tools. The comparative analysis shows that all performance metrics remain nearly the same in both execution environments. The simulation results show consistent behaviour across multiple platforms because they maintain identical performance across various testing environments.

VIII. THROUGHPUT ANALYSIS

The study examines throughput performance across three node densities: 20, 30, and 50. The average throughput values obtained are 195,157.6 bps, 195,182.4 bps, and 195,196.8 bps, respectively. The connections between nodes improve routing options, enabling better data transmission as the number of nodes increases.

The network maintains consistent performance, stable across different operational scales, with only slight fluctuations in its throughput measurements. The standard deviation values (±26.6, ±14.9, and ±20.5) show low values, which indicate that the simulation results maintain consistency throughout multiple testing sessions. The findings demonstrate that all results are highly consistent and produce identical outcomes across multiple testing sessions. Throughput performance shows identical patterns in both native Linux systems and virtualized environments which demonstrates that execution platform selection has no major impact on throughput results. The results demonstrate that throughput performance remains steady across varying node densities, confirming that the NS-2.35 simulator is independent of platform dependencies. Figure 7(a) provides a visual comparison of throughput performance between different systems.

A. End-to-End Delay Analysis

The study evaluates network latency using End-to-End (E2E) delay measurements collected across varying node densities. The average delay values observed in all scenarios remain very low, ranging from 0.005853 s to 0.005986 s. The delay measurements show only minor changes when additional nodes are introduced, demonstrating that changes in node density have no significant impact on delay. The routing protocol demonstrates the ability to manage packet transmission under heavy network traffic conditions according to this behaviour. The standard deviation values are extremely small (around ±0.00005), which confirms that the results are both stable and repeatable across multiple simulation runs. The simulation results show no difference in execution-platform latency between native Linux and virtualised environments. The network shows persistent low-latency performance throughout all testing scenarios. Figure 7(b) displays E2E delay

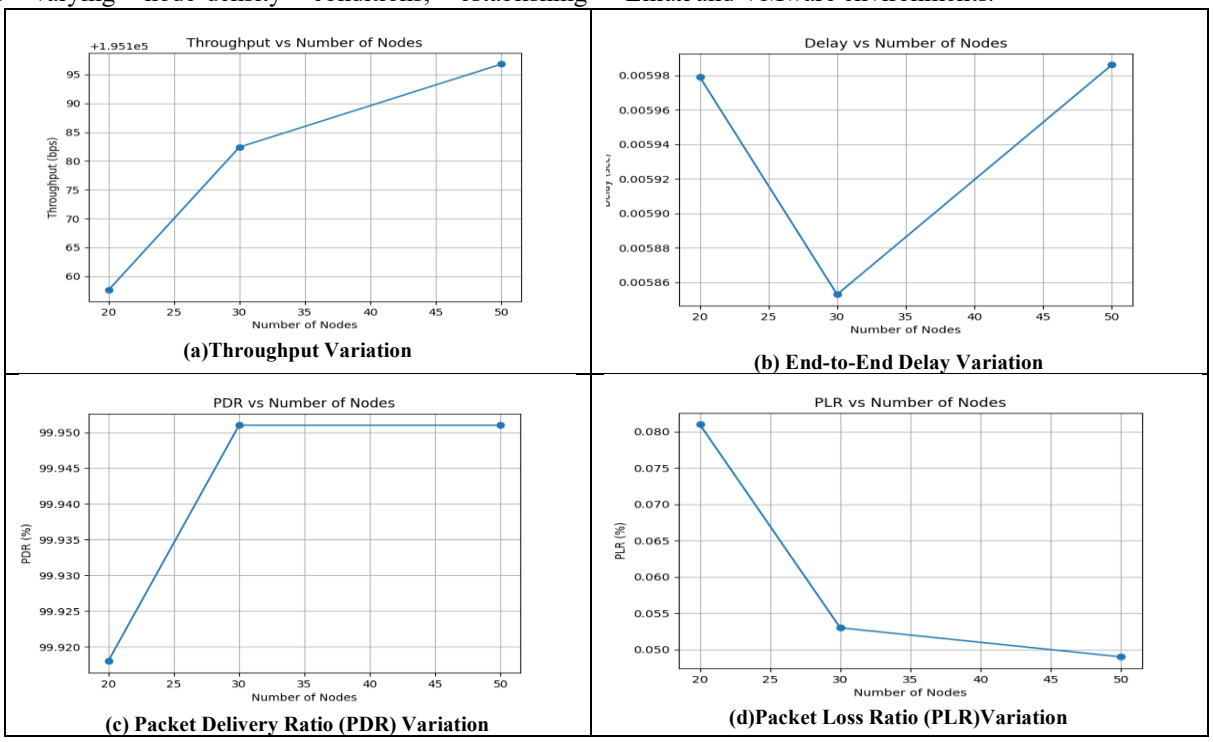
measurements conducted by researchers at various node-density levels.

B. Packet Delivery Ratio Analysis

The Packet Delivery Ratio (PDR) measures the network's success in delivering data packets to their intended recipients. The study results demonstrate that all tested scenarios produced high PDR values, averaging approximately 99.9% across testing. The network successfully transmits most packets because it uses efficient routing systems, which decrease packet loss. The standard deviation values (±0.03 to ±0.05) are very low, which confirms that the results remain stable and consistent across multiple simulation runs. The native Linux environment and the virtualised environment yield identical results, resulting in no detectable difference in PDR between them. The study results demonstrate that simulation outcomes are independent of the execution platform, while virtualisation does not affect packet delivery performance. The network maintains consistent performance across varying node-density conditions, establishing dependable operation. The PDR performance results for native Linux and VMware environments are shown in Figure 7(c).

C. Packet Loss Ratio Analysis

Packet Loss Ratio (PLR) measures the percentage of data packets lost during network transmission. The study results demonstrate that PLR values remain extremely low across all testing scenarios, ranging from 0.049% to 0.081%. The results show that the network successfully routes data with minimal loss, as only a tiny fraction of packets fail to reach their intended destination. The standard deviation values are also very small, reflecting the stability and consistency of the simulation results across different runs. The execution platform does not affect packet loss behaviour, as both native Linux and virtualised systems show similar PLR results. The simulation framework shows reliable performance because the packet loss rate stays constant while node density varies. Figure 7(a–d) shows the comparative results between native Linux and VMware environments.



(a)Throughput Variation

(b) End-to-End Delay Variation

(c) Packet Delivery Ratio (PDR) Variation

(d)Packet Loss Ratio (PLR)Variation

[Fig.7: (a–d): Performance Analysis of AODV under Different Node Densities (Source: Generated from Simulation Trace Files)]

D. Statistical Validation of Results

The study relies on statistical validation to verify that the simulation results remain accurate across multiple test runs. The research team performs multiple tests for each experimental condition and assesses the results using established statistical methods. The validation process establishes simulation model stability through testing different executions of the simulation. The summary of statistical validation for the performance metrics is presented in Table VII.

Table VII: Statistical Validation of Performance Metrics Using Mean and Standard Deviation (Source: Author's own Simulation results)

Nodes	Throughput (Mean ± SD) (bps)	Delay (Mean ± SD) (sec)	PDR (Mean ± SD) (%)	PLR (Mean ± SD) (%)	Runs
20	195,157.6 ± 26.6	0.005979 ± 0.00023	99.918 ± 0.05	0.081 ± 0.05	5
30	195,182.4 ± 14.9	0.005853 ± 0.00005	99.951 ± 0.03	0.053 ± 0.03	5
50	195,196.8 ± 20.5	0.005986 ± 0.00017	99.951 ± 0.04	0.049 ± 0.04	5

E. Multi-Run Analysis

The researchers ran five separate simulations for each of their three scenarios, with 20, 30, and 50 nodes, to reduce the effects of random mobility and traffic pattern changes on their testing results. The

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

simulation results showed consistent performance across all tests because the system-maintained operation throughout the testing period. The simulation setup yielded reliable results because there were no major differences across testing sessions. The multi-run method generates more trustworthy results because it demonstrates that multiple observations confirm the findings discovered in a single execution.

F. Mean ± Standard Deviation Evaluation

To quantitatively assess consistency, statistical measures such as the mean and standard deviation (Mean ± SD) are computed for all performance metrics.

The analysis reveals that the standard deviation values are extremely low across all scenarios:

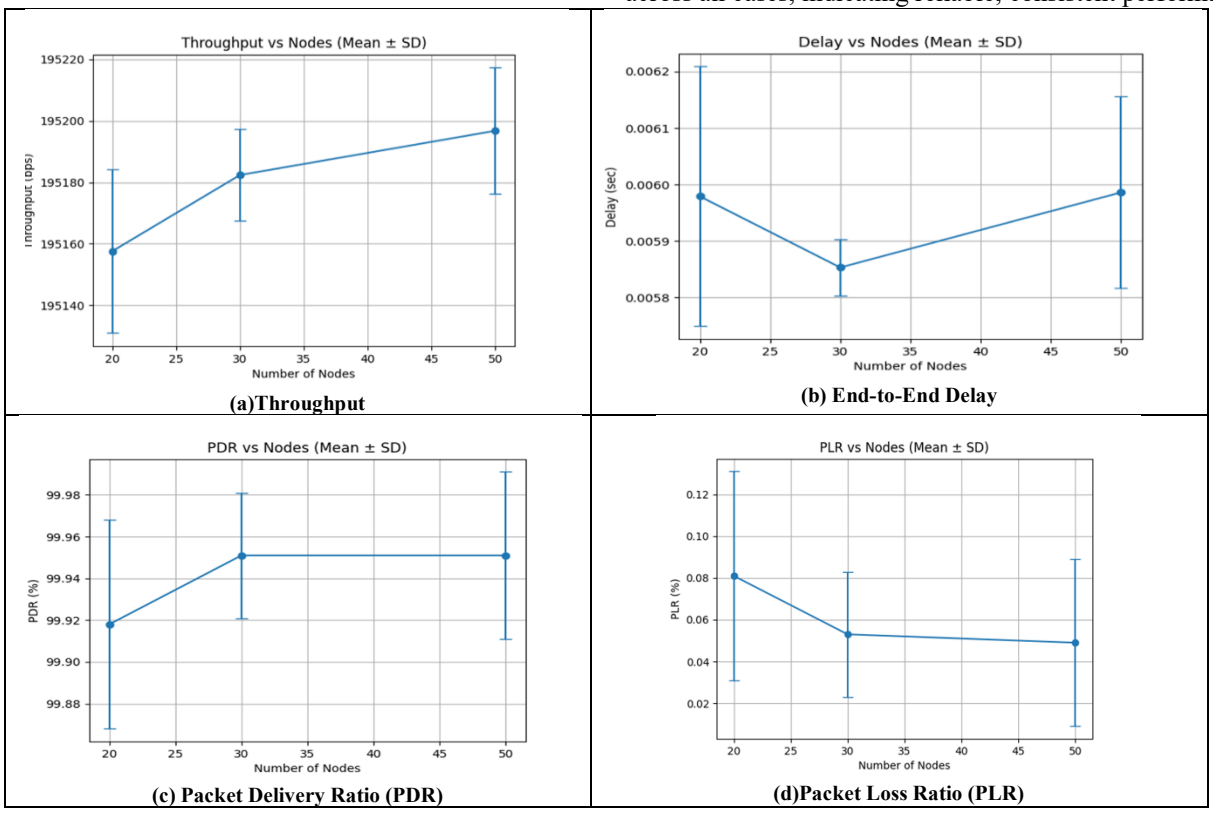
- Throughput: Very low variation (±14.9 to ±26.6)
- End-to-End Delay: Negligible variation (as low as ±0.00005)
- PDR: Minimal variation (±0.03 to ±0.05)
- PLR: Minimal variation (±0.03 to ±0.05)

These low standard deviation values indicate that the simulation results are highly consistent, with negligible dispersion around the mean.

The use of Mean ± SD provides a clear statistical representation of the stability and repeatability of the experimental outcomes.

G. Error Bar Representation and Interpretation

The performance metric distributions are shown using error bar graphs, with measurements based on the Mean ± Standard Deviation method. The graphs demonstrate consistent results across multiple simulation runs. The error bars remain extremely small in all scenarios, indicating that the sample values are close to the average. The simulation results demonstrate output stability, as identical results are observed across multiple experiments. The performance metrics exhibit low random variation because their execution conditions have no impact on the measurement results. The standard deviation (Mean ± SD) in Figure 8(a–d) is low across all cases, indicating reliable, consistent performance.



(a)Throughput

(b) End-to-End Delay

(c) Packet Delivery Ratio (PDR)

(d)Packet Loss Ratio (PLR)

[Fig.8: (a–d): Error Bar Analysis (Mean ± SD) of Network Performance Metrics Across Simulation Scenarios (Source: Generated from Simulation Trace Files)]

H. Cross-Platform Comparative Analysis

The section compares simulation results from native Linux and virtualised environments to examine how execution platforms affect simulation outcomes. The analysis uses NS-2.35 with identical configuration settings to guarantee an unbiased evaluation between both systems. The main objective is to examine whether virtualisation affects the accuracy, consistency, and reliability of performance metrics in MANET simulations.

I. Native vs Virtualized Performance Comparison

The comparative analysis of performance metrics shows that the results from native and virtualised environments are nearly identical across all simulation scenarios. The two setups display identical results for throughput, end-to-end delay, and packet delivery ratio (PDR) and packet loss ratio (PLR) measurements. The evaluation carried out at different node densities (20, 30, and 50 nodes) also shows similar trends, indicating that the simulation results remain stable across execution platforms. The performance metrics remain unchanged despite the slight variations in execution time and system resource consumption.

The findings demonstrate that native and virtualised environments produce

identical simulation results under the same experimental conditions.

J. Impact of Virtualization on Simulation Fidelity

The term simulation fidelity describes the extent to which simulation results match the established model and its input specifications. The study results demonstrate that virtualization technology does not create any measurable disturbance to the simulation results. The results obtained from virtualized environments remain consistent with those from native systems. The simulator's performance evaluation remains unaffected by system-level overheads introduced by virtualization. The results demonstrate that virtual machines enable users to create virtual environments that maintain simulation accuracy. The research study demonstrates that virtualised environments maintain simulation fidelity, making them trustworthy for network simulation research.

K. Deterministic Behaviour of NS-2.35

The results from multiple experiments testing different execution platforms show that NS-2.35 exhibits predictable behaviour. The simulator produces identical results when users input identical parameters together with configuration settings across multiple system environments. The statistical analysis further supports this observation, as only very small variations are observed in the results. The simulation results depend primarily on the established input parameters, which will not be affected by external system components. Simulation-based research requires deterministic behaviour because it establishes conditions that enable researchers to obtain reliable results that multiple platforms will reproduce.

L. Key Findings and Observations

The experimental analysis leads to several key observations:

i. The simulation results from native Linux and virtualized environments show identical results for all tested parameters, which include throughput, end-to-end delay, and packet delivery ratio (PDR) and packet loss ratio (PLR) testing.
ii. The simulation results show extremely low standard deviation values, which demonstrate that the results from multiple simulation tests maintain high consistency without any substantial differences.
iii. The network performance stays constant at various node densities, which include 20 nodes, 30 nodes, and 50 nodes, thus demonstrating the simulation system's ability to handle increased workload.
iv. The simulation results remain unchanged by virtualization, but execution time and system performance display minor differences.
v. The research demonstrates that NS-2.35 operates as a deterministic system that does not depend on specific platforms when researchers use identical configurations.

IX. DISCUSSION

The study results demonstrate that NS-2.35 simulation research maintains high reliability through its simulation-based research methods. The results show that simulation outcomes remain unchanged across different execution platforms, despite people believing that platform selection affects simulation results. The results show that virtualisation primarily affects execution speed and system performance, while the simulator's internal operations remain the same. Simulation experiments can use both native and virtualized platforms as their research environments. The study found that statistical validation operates as a significant element. The researchers reduced the effects of random variation by conducting multiple simulation runs and analysing the results using the mean and standard deviation. Repeated testing of the NS-2.35 system indicates that it operates according to predetermined patterns. The research tool enables researchers to run their experiments across various systems, producing consistent results.

X. CONCLUSION

The research conducted a comprehensive examination of the complete process, including installing NS-2.35, configuring it, and testing its performance on native Linux systems and virtualised systems. The team used a systematic approach to implement the simulator and assess its impact on simulation outcomes across different execution environments.

The experimental results demonstrate that essential performance metrics, including throughput, end-to-end delay, packet delivery ratio, and packet loss ratio, remain consistently unchanged across both test settings. The statistical analysis using mean and standard deviation also confirms that the results are stable and can be reproduced consistently.

The study has demonstrated that NS-2.35 maintains its operational behaviour across different platforms when identical simulation parameters are applied. The system base determines results only through user-configured input.

The study demonstrates that NS-2.35 is a reliable simulation tool for MANET studies. Virtualised environments allow users to maintain operational flexibility while achieving accuracy, although native Linux systems offer minimal performance benefits. Researchers can select their preferred platform based on their project needs and available resources.

The originality of this study lies in its cross-platform evaluation of NS-2.35, which confirms that the simulator delivers identical performance across native Linux and virtualised systems when executed under uniform conditions. This validation has not been clearly addressed in earlier literature. By establishing platform-independent reproducibility, the study enhances confidence in NS-2.35 as a dependable tool for MANET research, even when system configurations or deployment environments differ.

FUTURE WORK

Future research can extend this work by analysing the impact of virtualisation in more complex network scenarios that require testing across multiple network nodes and various traffic patterns. The study needs to assess both advanced routing protocols and hybrid routing protocols because this will determine whether the platform maintains its independence during active system changes.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Researchers can conduct performance tests using contemporary network simulators to assess their scalability and accuracy. Cloud-based platforms and containerised environments warrant further exploration because these technologies help researchers assess their impact on simulation speed and their ability to enable reproducible experiments across distributed systems.

DECLARATION STATEMENT

As the article's author, I must verify the accuracy of the following information after aggregating input from all authors.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** This article has not been funded by any organisations or agencies. This independence ensures that the research is conducted objectively and without external influence.
- **Ethical Approval and Consent to Participate:** The content of this article does not necessitate ethical approval or consent to participate with supporting documentation.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Author's Contributions:** The authorship of this article is contributed equally to all participating individuals.

REFERENCE

1. Abdullah, A.M., An efficient approach for minimizing control packets and enhancing route stability in mobile ad-hoc networks. The Journal of Supercomputing, 2026. 82(5): p. 305. https://doi.org/10.1007/s11227-026-08480-y
2. Irani, F., VSIM: a new simulation and performance evaluation tool for MANET and VANET. International Journal of Information Technology, 2024: p. 1-17. https://doi.org/10.1007/s41870-024-02223-z
3. Shakya, A., S. Ali, and P. Nand. Simulation-Based Performance Analysis of MANET Routing Protocols: Proactive vs Reactive, in 2025 Optical Communication, Photonics, Telecommunications, and Intelligent Machine Applications (OPTIMA). 2025. IEEE. https://doi.org/10.1109/OPTIMA67660.2025.11380316
4. Jain, S.M., Linux Containers and Virtualization. A Kernel Perspective, 2020: p. 2020-10. https://link.springer.com/book/10.1007/978-1-4842-9768-1
5. Selim, I.M., et al. (2025) – MANET Routing Protocols' Performance Assessment under Dynamic Network Conditions, Applied Sciences, 15(6): 2891. https://doi.org/10.3390/app15062891
6. Smera, C. and J. Sandeep. Networks simulation: Research-based implementation using tools and approaches. In 2022, the IEEE 3rd Global Conference for Advancement in Technology (GCAT). 2022. IEEE. https://doi.org/10.1109/GCAT55367.2022.9972119
7. Suutari, L., Virtualized learning environment for network security using infrastructure as code practices. 2024, L. Suutari. https://urn.fi/URN:NBN:fi:oulu-202412177352
8. Dorathy, I. and M. Chandrasekaran, Simulation tools for mobile ad hoc networks: a survey. Journal of applied research and technology, 2018. 16(5): p. 437-445. https://www.scielo.org.mx/scielo.php?pid=S1665642320180005004370&script=sci_arttext&tlng=en
9. Behura, A., A. Kumar, and P.K. Jain, A comparative performance analysis of vehicular routing protocols in intelligent transportation systems. Telecommunication Systems, 2025. 88(1): p. 26. https://doi.org/10.1007/s11235-024-01243-1
10. Bagirathan, K., N. Saravanan, and K. Vijayabhaskar, An intelligent recurrent neural network-driven secured routing protocol for vehicular ad hoc networks. Knowledge-Based Systems, 2025. 317: p. 113371. https://doi.org/10.1016/j.knosys.2025.113371
11. Collins, A.J., F. Sabz Ali Pour, and C.A. Jordan, Past challenges and the future of discrete event simulation. The Journal of Defence Modelling and Simulation, 2023. 20(3): p. 351-369. https://doi.org/10.1177/15485129211067175
12. Lee, C., H. Kwon, and Y.I. Lee, OPC UA-based three-layer architecture for aggregated microgrids integrating edge cloud computing and IEC 62264. Journal of Industrial Information Integration, 2025: p. 100965. https://doi.org/10.1016/j.jii.2025.100965
13. Williams, M., Coordinated Land Use and Transportation Planning–A Sketch Modelling Approach. 2010. . http://hdl.handle.net/1807/25512
14. Hess, K., Practical Linux system administration: a guide to installation, configuration, and management. 2023: " O'Reilly Media, Inc.". https://books.google.co.in/books?hl=en&lr=&id=UVG6EAAAQBAJ&oi=fnd&pg=PT10&dq=Hess,+K.,+Practical+Linux+system+administration:+a+guide+to+installation,+configuration,+and+management.+2023:+%22+O%27Reilly+Media,+Inc.%22.&ots=PYoKX6PyuS&sig=rdoiVNoJFJz05Yq2vTuZaKfLbpU&redir_esc=y#v=onepage&q&f=false
15. Rozehkhani, S.M., F. Mahan, and W. Pedrycz, Efficient cloud data centre: An adaptive framework for dynamic Virtual Machine Consolidation. Journal of Network and Computer Applications, 2024. 226: p. 103885. https://doi.org/10.1016/j.jnca.2024.103885
16. Yoginath, S.B. and K.S. Perumalla. Optimized hypervisor scheduler for parallel discrete event simulations on virtual machine platforms. in SimuTools. 2013.
17. Kumar, S. and S. Das, Using Ansible Playbooks to Port Non-cloud-Native Applications Across Linux Distributions: A Novel Approach. SN Computer Science, 2023. 4(5): p. 644. https://doi.org/10.1007/s42979-023-02030-6
18. Ma, Y., et al., A survey of DDoS attack and defence technologies in multi-access edge computing. IEEE Internet of Things Journal, 2024. 12(2): p. 1428-1452. https://doi.org/10.1109/JIOT.2024.3490897
19. Hemanand, D., et al., A secure AODV algorithm based on malicious nodes detection in flying ad-hoc networks. International Journal of System Assurance Engineering and Management, 2026: p. 1-10. https://doi.org/10.1007/s13198-026-03272-2
20. Khan, H., et al. Performance Evaluation of DSR, AODV and MP-OLSR Routing Protocols Using NS-2 Simulator in MANETs. in International Conference on Cognitive Computing and Cyber Physical Systems. 2023. Springer. https://doi.org/10.1007/978-3-031-48891-7_10

AUTHOR'S PROFILE

Mr. Vinay Sahu is an Assistant Professor at LNCT, Bhopal, and is currently pursuing his PhD in Computer Science and Engineering at Rabindranath Tagore University (RNTU), Raisen. He has completed his M.Tech from MANIT Bhopal and holds a B.E. (Hons.) degree from Barkatullah University, Bhopal. His research interests include Routing Protocols and Mobile Ad Hoc Networks (MANETs). He has contributed to several research publications in reputed journals and conferences.

Dr. Ankur Khare, Assistant Professor, Department of Computer Science and Information Technology, Rabindranath Tagore University, Raisen, Madhya Pradesh, India. He is an academician and researcher specializing in Artificial Intelligence, Machine Learning, and Network Security. He has contributed to several research publications in reputed journals and conferences. His research interests include data analytics, intelligent systems, and advanced machine learning techniques. He is committed to academic excellence and fostering innovative learning among students.

Dr. Rani Sahu is an Associate Professor in Computer Engineering & Applications at IES College of Technology, Bhopal, Madhya Pradesh, India, with over 16 years of academic and research experience. She received her PhD in Computer Networking from MANIT, Bhopal. Her research interests include Mobile Ad Hoc Networks (MANETs), energy-efficient routing, reinforcement learning, and machine learning. She has published 20+ research papers in reputed journals, IEEE conferences, and book chapters, and actively contributes to advanced research in networking and intelligent systems.