2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT) | 979-8-3503-6944-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/DCOSS-IoT61029.2024.00079

Development of a Mobile Ad-Hoc Network Testbed: Modular Implementation of Ad-Hoc On-Demand Distance Vector Routing

Gage Gailbreath, Andre Koka, Mohammed Gharib, Alireza Ebrahimi, Fatemeh Afghah
Holcombe Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, USA
E-mail:{ggailbr, arkoka, alghari,alireze, fafghah}@clemson.edu

***Abstract*—It is widely acknowledged that real implementation stands out as the most accurate performance evaluation technique, surpassing simulation-based and analytical-based approaches. However, the inherent challenges and costs associated with real implementation pose significant barriers to its widespread adoption. While several testbeds have successfully navigated these challenges for ad-hoc communication, existing solutions often come with drawbacks, such as high costs, dedication to specific protocols or hardware, and limitations in terms of mobility and AI compatibility. This paper introduces a novel, versatile testbed designed for the performance evaluation of mobile ad-hoc networks (MANETs). Noteworthy for its cost-effectiveness, modularity, mobility, and AI compatibility, this testbed supports non-homogeneous nodes and operates seamlessly on any system with a Linux operating system (OS). It is complemented by a user-level Application Programming Interface (API) that facilitates the implementation of various protocols on the testbed by separating the implementation from the OS functionality[1]. To demonstrate its practicality, we implemented the Ad hoc On-Demand Distance Vector (AODV) routing protocol, a complex and well-known example, and compared its results with those obtained from the network simulator ns-3.**

***Index Terms*—Testbed, MANET, AODV, Routing Protocol, Wireless Communications.**

I. Introduction

In communication systems, wireless ad-hoc networks operate on a communication topology devoid of infrastructure, unlike more conventional network structures that rely on deployed infrastructure, such as cellular networks. While infrastructure-based topologies typically exhibit superior performance, the importance of ad-hoc networks becomes paramount in situations where infrastructure is non-existent, compromised, or impractical to establish. Such networks are especially vital in mission-critical scenarios, including military operations in hostile environments and natural and man-made disasters. These environments, where rapid deployment and flexibility are key, highlight the indispensable role of ad-hoc networks.

One specific category of ad-hoc network is the Mobile Ad-Hoc Network (MANET), wherein nodes possess unrestricted mobility in three-dimensional space and communicate with each other in a peer-to-peer manner. MANETs hold particular significance in the realm of Internet-of-Things (IoT) applications. The U.S. Army's unmanned aircraft system roadmap for the years 2010-2035 explicitly underscores the strategic priority of considering infrastructure-free networking [1].

On one hand, the performance of each algorithm proposed to be used in MANET is required to be carefully evaluated, where the most reliable and accurate method for evaluating performance is through real-world measurements. Such measurements provide concrete insights into how the algorithms will function in actual MANET environments, where variables and conditions can be considerably different from theoretical or simulated scenarios. On the other hand, enabling communication in MANETs demands the use of sophisticated algorithms to discover, maintain, and update routes within the network as nodes dynamically relocate [2]. Deploying these complex algorithms into a physical system is particularly challenging, as it involves modifying the networking stack of the node's underlying operating system to establish and utilize routes. Such deployment requires not only a deep understanding of the operating system's architecture but also a meticulous approach to ensure that the newly integrated algorithms interact seamlessly with the existing system components, all while maintaining the stability and efficiency of the network. Factors including different operating systems, OS updates, and signal propagation characteristics contribute to the complexity of physical MANET network implementations. One approach to simplify this process is to implement such algorithms at the application layer, using techniques like socket programming. While this pragmatic solution significantly reduces implementation complexity compared to lower-layer approaches, it introduces a considerable delay in basic network operations, as demonstrated in Section (IV).

In response to the mentioned challenges, this paper introduces a MANET testbed designed to facilitate physical implementations of MANET networks. The testbed offers a user-friendly API that interfaces with the Linux operating system entirely from the user-space, leveraging built-in Linux modules such as Netfilter and Netlink, along with the Linux command-line utility iptables. This testbed serves as a practical platform, enabling the real-world testing and refinement of MANET algorithms and configurations, thereby bridging the gap between theoretical research and tangible application in MANET environments. Incorporating a robust design, the API within this testbed heavily leverages the Linux modules mentioned earlier, employing them for tasks such as sending,

This material is based upon work supported by the National Science Foundation under Grant Numbers CNS-2120485, CNS-2318726 and CNS-2232048.

[1]The project is publicly available at https://github.com/ggailbr/MANET-Testbed

2325-2944/24/$31.00 ©2024 IEEE
DOI 10.1109/DCOSS-IoT61029.2024.00079

capturing, queuing, and filtering packets as defined by the routing protocol. This involves the modification of Linux routing tables and seamless communication between the user-space and kernel-space through Netlink. Section (III) delves into the intricacies of the MANET testbed's design, highlighting the API functions and their diverse capabilities.

As discussed in Section II, prior work primarily focused on non-generalized implementations, rendering the reuse of these designs for new performance evaluations either impractical or prohibitively expensive. Furthermore, many implementations resorted to manual restrictions through firewalls and packet-dropping rules to limit node connections in testing scenarios which is considered a limitation of physical implementations and deviates from real-world scenarios. While several MANET testbed implementations exist in the literature, they often lack modular designs, leading the test to be impractical for the implementation of alternative protocols [3]–[10]. Even when modularity is present, they frequently fall short in providing users with an API or other tools to create fully customized routing protocols for use with the testbed. This paper contributes to the field by presenting a modular MANET testbed that remains agnostic to both the routing protocol under test and the underlying Linux OS. The testbed is not only portable and distributable as a software package but also offers users an API to implement custom MANET protocols. This versatility empowers researchers to operate heterogeneous MANETs comprising various node types, showcasing both mobility and compatibility with AI. The testbed stands out for its cost-effectiveness, as the APIs can be employed on any devices with Linux OS, which are available at a low cost, starting as low as $20 per node.

To validate the efficacy and practicality of the testbed, this study includes the implementation of the Ad-Hoc On-Demand Distance Vector (AODV) routing protocol using the provided APIs, on several Raspberry PI 400 nodes. AODV, a sophisticated MANET routing protocol outlined in RFC 3561, heavily relies on control-plane messages circulating throughout the MANET network. Despite the existence of other MANET routing protocols that surpass AODV's performance, AODV is still considered a benchmark general-purpose route discovery algorithm. Hence, it serves as an apt choice for evaluating the MANET testbed's functionality. Although further extensive testing is warranted, initial qualitative results affirm the operational efficiency of the testbed, exemplified by the successful implementation of AODV through the API functions. Additionally, to underscore the significance of real-world implementations, we replicated the same scenarios on the ns-3 network simulator for comparative analysis. This comparison illuminates how performance evaluation results, even for seemingly simple scenarios, can vary between simulations and real-world implementations. Key performance indicators (KPIs) such as routing traffic, route discovery time, network throughput, and the time required to transfer a large file in the network were considered pivotal for this evaluation [3], [11].

II. Related Work

To evaluate the performance of the MANET protocols, there has long been a recognized need to move beyond the realm of simulation. As such, several other previous works have used the idea of a testbed to evaluate MANET networks in various forms. We discuss these previous works in three different categories, testbeds developed for a specific protocol, modular testbeds, and application-specific testbeds.

Testbeds for Specific Protocols: Previous research has largely focused on the implementation and evaluation of specific MANET protocols, often overlooking the potential for modularity in a MANET testbed. In [9], Maltz et al. creates an ad-hoc testbed to implement the dynamic source routing (DSR) protocol, sharing their experiences in the creation of their DSR implementation. This testbed provides impressive features including seamless internet integration, but does not focus on modular routing protocol implementation, and provides no API to make routing protocol implementation simpler for a user. In [12], Brown et al. create a modular MANET testbed and use it to evaluate another DSR implementation, evaluating latency and network traffic. This testbed uses the Click Modular Router to implement routing protocols modularly, but requires a Click implementation of a particular routing protocol to be used with the testbed. Weber et al. present a novel testbed in [10] that deploys stationary ad-hoc testbed nodes within public spaces in Dublin, which can be further improved with mobile nodes like laptops and smartphones. This work provides no actual evaluation, however, of any routing protocols and provides relatively little detail regarding the design and use of the testbed. Zola et al. implement AODVv2, which is a well-known optimization of AODV, on ARM-based devices in [8]. They found that their implementation of AODVv2 was fully operational and intend to provide their code to the research community, but does not consider any other routing protocols besides AODVv2. Finally, Biagioni presents a testbed implementation of the All-net routing protocol in [7], evaluating All-net on metrics such as transmission time and success rate. Biagioni's work does not include modular routing protocol implementation and lacks a rigid evaluation scenario.

Modularly Implemented Testbeds: The need for modular implementation of various ad-hoc protocols in a robust MANET testbed is well-recognized. In [13], Maltz et al. provides quantitative results for the same DSR implementation that is discussed in [9]. The testbed developed for DSR implementation is designed with intention to facilitate testing of additional routing protocols in the future. However, it currently lacks an API that would allow users to implement their own custom routing protocols. In [14], Nordstrom et al. present a powerful testbed that takes advantage of kernel and user-space Linux applications to provide a modular and interactive testbed for users. Despite the powerful capabilities of the present "test choreography" scripts in [14], the user is still not given complete freedom to develop a fully-custom protocol, as the test choreography scripts are limited in terms of the number of actions they support. In [15], Biswas et al. present a testbed for virtual ad-hoc networks, which takes advantage of emulated networks that behave like real wireless networks. However, such work still relies on simulation, which cannot accurately reflect real-world wireless networks, making

TABLE I: An Overview of Related Works

Related Work	Modular Protocols	General Purpose	Specialization	Hardware-based	Node Cost	Heterogeneous Node Support	Mobility	AI Support	Drawbacks
Maltz et al. [9]		✓	N/A	✓	$200		✓	✓	Difficult to implement multiple protocols
Brown et al. [12]	✓	✓	N/A	✓	$60		✓	✓	Requires a Click Monitor Router protocol implementation
Weber et al. [10]		✓	N/A	✓	Unknown	✓		✓	Requires external hardware to enable limited mobility
Zola et al. [8]			Benchmark for AODVv2 on ARM devices	✓	$500	✓		✓	Lack of mobility and modular routing protocols
Biagioni [7]			Testing AllNet routing protocol	✓	$25	✓	✓	✓	Only supports AllNet protocol
Maltz et al. [13]	✓	✓	N/A	✓	$30		✓		Does not provide API for custom routing protocols
Nordstrom et al. [14]	✓	✓	N/A	✓	$500	✓		✓	Does not provide API for custom routing protocols
Biswas et al. [15]	✓		Virtual Ad-Hoc Network (VAN) testbed		Unknown	✓		✓	Focuses primarily on simulation
Plestys et al. [16]	✓	✓	N/A	✓	$146				Uses physical cables to emulate wireless connections
Hussain et al. [17]	✓	✓	N/A	✓	$250	✓	✓		Does not provide API for custom routing protocols
Tabrizi [18]	✓	✓	N/A	✓	$2,100				Difficult to add new routing protocols
Karygiannis [19]	✓	✓	N/A	✓	Unknown				No actual testbed implementation provided, only design
Desai [20]	✓		video transmission	✓	$2,100				
Panaousis et al. [6]			security analysis of OSLR protocol	✓	$500			✓	Lack of mobility and modular routing protocols
Muchtar et al. [5]			Investigates best robots for testbed mobility	✓	N/A		✓		No actual testbed development presented
Muchtar et al. [4]			N/A	✓	N/A		✓		No actual testbed development presented
Bouachir et al. [3]			AI for drone swarm communication	✓	Unknown		✓	✓	
This Work	✓	✓	N/A	✓	$20	✓	✓	✓	Requires static IPv4 addresses for nodes

[15] an imperfect representation of physical MANET implementations. Plestys and Zakarevicius present a unique modular testbed in [16] which employs physical cable connections that imitate real wireless connections through precise cable attenuation. Despite this approach, the reliance on cable attenuation does not fully replicate a genuine physical MANET, and [16] still provides no API to users for the creation of routing protocols. Karygiannis and Antonakakis discuss an in-progress MANET testbed in [19], which utilizes mNet, mDog, and mSignal tools to create arbitrary network topologies and change transmission powers in a physical MANET network, rather than a simulation. However, [19] provides no evaluation results or routing protocol implementations, only the design of the testbed that has yet to be used for experimental results. In [18], Tabrizi uses Software-Defined Radios (SDRs) to create a flexible and reconfigurable MANET testbed, which is then used to implement open-source versions of Optimized Link State Routing Protocol (OLSR) and Babel routing protocols and evaluated them between each other. Tabrizi's testbed has the novel ability to support open-source implementations of various routing protocols. However, it does not provide the portability feature and lacks an API, unlike our proposed work. [20] implements a testbed for multi-hop wireless ad-hoc networks which uses SDR to implement live video feed. Unlike our paper, [20] does not discuss the modular implementation of different ad-hoc routing protocols. Finally, Hussain et al. [17] discuss their design for a MANET testbed that has similar advantages to our work. Like our work, Hussain et al. implement the testbed fully in user-space, and can test routing protocols modularly, but do not offer an API to the user for the creation of fully custom routing protocols for use with their testbed.

Application-Specific Testbeds: The last section of this literature review considers recent works that have used MANET testbeds and communication protocols and further specialized them with other fields. In [6], Panaousis et al. use a testbed implementation of OLSR to evaluate various cryptographic schemes that can be used to secure the MANET network as a whole. They demonstrated the variations of several communications metrics such as bit rate as a function of encryption protocols, but do not focus on testbed development and modular implementation. In [5], Muchtar et al. perform an investigation of which physical robots are best used in physical MANET networks for providing mobility. Similarly, Muchtar et al. [4] investigated different MANET testbeds that use robots to provide mobility and discussed whether or not the mobility provided by these robots is successful and necessary. Lastly, Bouachir et al. [3] create a testbed for use with multi-drone systems utilizing deep learning to meet the quality-of-service demands of all drones. Their work is a powerful MANET network that has specialized in deep learning but does not extend to facilitating broader research and implementation in the MANET domain. Table (I) summarizes the state-of-the-art and compares them with our developed testbed.

III. Design and Implementation

To the best of our knowledge, this testbed stands out as the first modular MANET testbed developed on Linux kernel 5.14 or newer, enabling the utilization of Linux kernel modules not previously included by default. The design of the MANET testbed was influenced by various considerations. While the initial inspiration was drawn from the first implementation of AODV-UU [21], the widely adopted physical implementation of AODV for Linux devices, our MANET testbed operates entirely in user space. This design decision offers several advantages; i) User-space development facilitates the use of common C libraries and network programming; ii) Debugging user-space code is notably more straightforward compared to kernel development; iii) User-space development enhances portability and abstracts the routing protocol from the underlying OS, aligning with a key design goal; iv) Deploying code updates is streamlined in user space, eliminating the need to recompile the Linux kernel on multiple nodes. However, user-space development has its drawbacks as well. The most notable disadvantage is the need to queue network traffic, transfer packets to user space, and then wait for a verdict to be issued for the packet. This manual packet handling is less efficient than the operating system's packet handling capabilities. In this section, we review the design and implementation of the testbed from three perspectives, the Linux modules, the developed APIs, and the implementation of the AODV routing protocol for testbed verification.

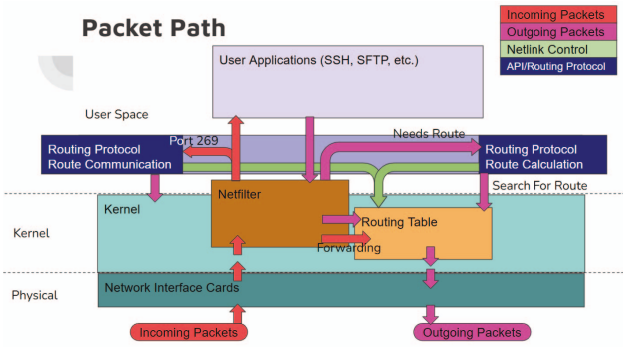


Fig. 1: MANET Testbed Architecture

A. Linux Modules

Due to MANET testbed development remaining in user-space, a way to communicate with the Linux kernel was mandatory. In addition, the MANET testbed needs to perform actions like modifying routing tables, queuing packets, and retrieving wireless interface information while still providing the user control over when these tasks are completed. To accomplish this, the MANET testbed takes advantage of several Linux kernel modules that are standard for any Linux installation of kernel 5.14 or newer. Table II shows the Linux modules or applications that were used. Each module's interaction with the Linux kernel and Linux networking stack is also shown in Figure 1.

TABLE II: Linux kernel Modules Required for MANET Testbed Operation

Linux Module or Tool	Purpose
UDP Sockets	Used to communicate between nodes, allowing the user to send control plane messages as needed.
Netlink	Used to communicate between kernel and user space. Important for retrieving a specific node's IPv4 address, or any other wireless interface information.
RTNetlink	An extension of Netlink, used exclusively to retrieve or modify the kernel routing table.
Netfilter	Used to filter packets and eventually send them to user-space. To avoid creating a custom queue structure, the `libnetfilter_queue` library was utilized.
iptables	Used to establish packet filtering rules and queue packets as needed by the testbed.

B. Developed API

To use the MANET testbed, the user is provided with a simple API consisting of nine currently implemented functions. While internal implementation of these functions involves the use of smaller helper functions, the entirety of the MANET testbed's functionality can be found within the main developed API functions. The MANET testbed is designed to abstract the OS details away from the user. Therefore, each API function implements some kind of OS communication and completes the appropriate task, with the user providing nothing but the appropriate inputs. For example, the implementer of the routing protocol can use the `AddUnicastRoutingEntry()` function to modify the Linux routing table with a new route, rather than creating, formatting, and sending a Netlink message as part of the routing protocol implementation code. The nine API functions each play a pivotal role in providing the testbed with enough functionality to implement multiple different MANET routing protocols. These functions are:

- **InitializeAPI()** - Performs some required setup for the library, and must be the first function called by the user.
- **AddUnicastRoutingEntry()** - Adds a unicast route to the main Linux routing table using Netlink.
- **DeleteEntry()** - Deletes a route from the main Linux routing table using Netlink.
- **SendUnicast()** - Sends a unicast message to a given destination using Linux User-Datagram Protocol (UDP) sockets.
- **SendBroadcast()** - Broadcasts a message to the entire MANET network using Linux UDP sockets.
- **GetInterfaceIP()** - Retrieves the Internet Protocol (IP) address of a particular interface using Netlink.
- **RegisterIncomingCallback()** - Registers the provided function as the callback function for queued incoming packets using Netfilter.
- **RegisterOutgoingCallback()** - Registers the provided function as the callback function for queued outgoing packets using Netfilter.
- **RegisterForwardCallback()** - Registers the provided function as the callback function for queued forwarded packets using Netfilter.

An advantage of using this high-level API is that it allows the routing protocol to be detached from how it interacts with the hardware. This abstraction allows upgrading and optimization independent of each other. A similar API can then be made in kernel space or a simulator and the routing protocols would not need to be adjusted. This benefits future protocols as they will not need to be rewritten by each researcher and unifies the platforms they are working with. If a researcher wants to focus on the optimization of the API, they can use the already written protocols as a point of comparison. Additionally, researchers can compare multiple protocols, written using the same API, without having to re-implement the protocols themselves, unifying the comparisons.

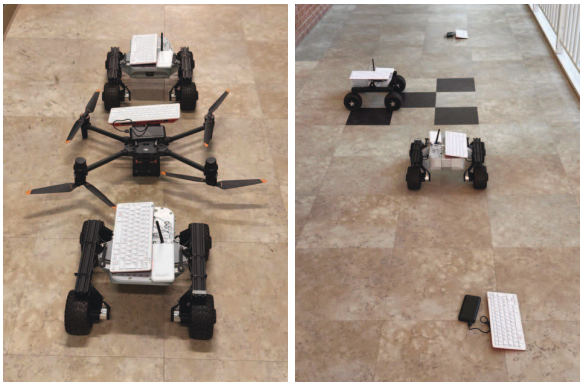
C. AODV Implementation

AODV is a standard routing protocol that has been used as a benchmark to evaluate other MANET routing protocols. To prioritize assessing the implemented testbed, a stripped version of AODV was created. AODV is a reactive routing protocol, meaning it only performs route discovery when a packet needs to be sent. When an outgoing packet is received, it is processed in the order of Algorithm (1).

For each node nearby in the network, it will receive the RREQ and either: forward it if the TTL allows, send back a RREP if it is the destination, or send back a RREP if it has a valid route to the destination. This process ensures traffic is only created when establishing routes and maintaining active routes. Nodes do not know the whole route themselves, only the next valid hop towards the destination. The operations of AODV are implemented using the API described in Section (III-B). This provides a protocol to assess the testbed and the performance of its various functions.

Algorithm 1: AODV

A route toward destination node is needed
if Current and active route exists **then**
  Send the packet from the path.
else
  condition ← true
  Create a Route Request packet (RREQ)
  TTL=1,{Time-To-Live (TTL)}
  Route Discovery():
  while condition **do**
    Broadcast RREQ
    TTL++
    if reached(maximum TTL) **or** Received (RREP) **then**
      condition ← false,{Route Reply (RREP)}
    end if
  end while
  Drop or send the packet depending on the result of the route discovery
end if



(a) (b)

Fig. 2: Different Test Scenarios

IV. Test Scenarios

To evaluate both the testbed and the AODV implementation, we conducted multiple test scenarios involving varying node counts, mobility patterns, and data transfer scenarios, as illustrated in Fig. (2). These scenarios were also replicated in the ns-3 network simulator for comparative purposes. While we used Raspberry PI 400 to test our testbed, Table (III) represents the ns-3 simulation parameters. Our findings indicate that the real implementation is notably more sensitive to environmental conditions, even in free space, in contrast to simulation results. To ensure a fair comparison, we established a multihop scenario with nodes aligned in a straight line. Each adjacent node is within communication range of one another but out of range of the next farther node, as depicted in Fig. (3). For each scenario, we increment the nodes by one and initiate a file transfer of 10^6 Bytes from the first node (n_0) to the last node (n_i).

In scenarios involving more than two nodes, we introduced mobility tests. Specifically, we moved the second node (n_1) upward with a fixed velocity, concurrently moving the third

TABLE III: Simulation Setting

Simulator version	ns-3 3.37
Number of nodes	[1 6]
Transmission power	7.5 dBm
File Size	10^6 Bytes
Packet Size	1448 Byte
Transmission Rate	72.2 Mbps
Speed range	$[0 \quad 0.4] m/s$
Traffic type	TCP NewReno
Wireless communication standard	IEEE 802.11n
Propagation loss model	Free-space propagation loss
Propagation delay model	Constant speed propagation delay

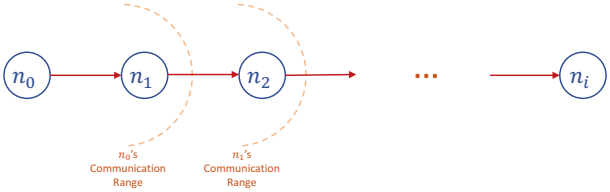
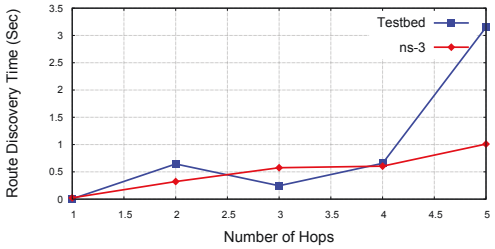


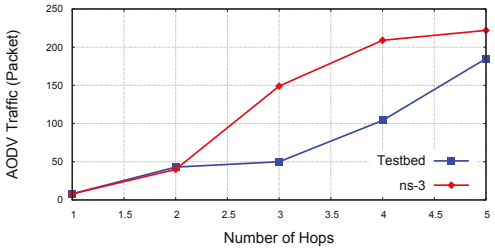
Fig. 3: Network Topology

node (n_2) toward n_0. This action leads to a temporary disconnection and a reroute from $n_0 \rightarrow n_1 \rightarrow n_2$ to $n_0 \rightarrow n_2$, i.e. restoring the communication route to the destination. To make the mobility smooth and steady, we utilized Leo Rovers, as represented in Fig. (2b).

We monitored all communications in the testbed using the *tcpdump* tool and saved the outputs as pcap files. The same procedure was followed in the simulations to ensure consistency of results. We evaluated four key parameters for both the simulations and testbed implementations, encompassing two factors specific to the AODV routing protocol and two representing general network metrics. For AODV, we measured the route discovery time, which represents the time duration from the initial file transfer attempt to the establishment of a

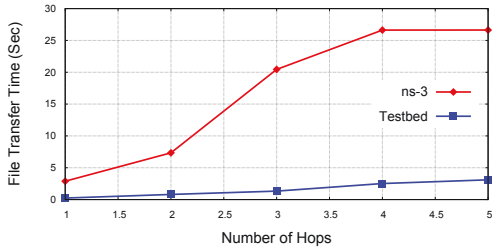


(a) Route Discovery Time

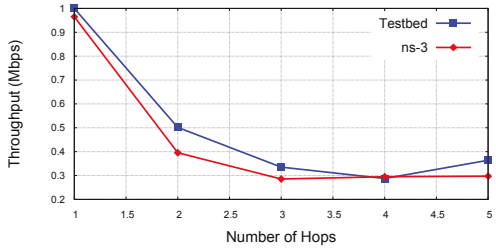


(b) Routing Traffic

Fig. 4: AODV Performance Evaluation



(a) File Transfer Time



(b) Network Throughput

Fig. 5: Network Performance Evaluation

route. Additionally, we assessed routing traffic, quantified as the number of AODV packets exchanged within the network to determine the optimal path. Regarding network parameters, we measured the time required for transferring an entire file from the source node to the destination and determined the network throughput for this file transfer operation.

Fig. (4a) compares AODV route discovery times between the simulation and the testbed, while Fig. (4b) illustrates the corresponding traffic. It is important to note that our objective is not to draw definitive conclusions about superiority of AODV implementation in the testbed versus ns-3, as results are contingent on various factors such as hardware specifications and environmental conditions. Our primary goal is to showcase the capability of the developed testbed in implementing diverse network protocols. We would also like to emphasize that, while simulations offer valuable insights, they have limitations in replicating real-world scenarios as real-world experiments depend on numerous factors that simulations may simplify or overlook for the sake of simplicity or generality.

Finally, Fig. (5a) illustrates the time required for the entire file transfer from the source node to the destination in scenarios with varying numbers of intermediate hops. Additionally, Fig. (5b) depicts the network throughput for the file transfer process. In addition to the mentioned scenarios, we implemented a basic version of AODV using Python socket programming. Although the results are not presented here, we observed that the route discovery time using socket programming is an order of magnitude higher than that required by the developed API, which is illustrated in Fig. (4a).

V. Conclusion and Future Work

This paper presents a MANET testbed, which boasts the ability to implement any MANET protocol on Linux devices running kernel 5.14 or newer while abstracting the operating system details away from the implementer of the routing protocol through user-friendly API. This work also uses a custom implementation of AODV to evaluate the testbed. Results show that all API functions are fully operational. This work can be extended in several ways, but the most important is to implement more MANET protocols with the testbed, implement a mobility manager for testbed nodes, and perform comparisons between the real implementation of protocols and their simulated versions, through tools such as ns-3.

References

[1] U.S. Army UAS Center of Excellence "Eyes of the Army", "U.s. army unmanned aircraft systems roadmap 2010-2035," Tech. Rep. ATZQ-CDI-C, 2010. [Online]. Available: https://fas.org/irp/program/collect/uas-army.pdf

[2] A. Rovira-Sugranes, A. Razi, F. Afghah, and J. Chakareski, "A review of ai-enabled routing protocols for uav networks: Trends, challenges, and future outlook," *Ad Hoc Networks*, vol. 130, p. 102790, 2022.

[3] O. Bouachir, M. Aloqaily, F. Garcia, N. Larrieu, and T. Gayraud, "Testbed of qos ad-hoc network designed for cooperative multi-drone tasks," ser. MobiWac '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 89–95.

[4] M. Farkhana and A. Abdul Hanan, "Mobility in mobile ad-hoc network testbed using robot: Technical and critical review," *Robotics and Autonomous Systems*, vol. 108, pp. 153–178, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889018302458

[5] F. Muchtar, H. Abdullah, M. H. Abd Wahab, R. Ambar, H. Hanafi, and S. Ahmmad, "Mobile ad hoc network testbed using mobile robot technology," *Journal of Physics: Conference Series*, vol. 1019, 06 2018.

[6] E. A. Panaousis, G. Drew, G. P. Millar, T. A. Ramrekha, and C. Politis, "A testbed implementation for securing OLSR in mobile ad hoc networks," *CoRR*, vol. abs/1010.4986, 2010.

[7] E. Biagioni, "A network testbed for ad-hoc communications using raspberry pi and 802.11," 01 2019.

[8] E. Zola, I. Martin-Escalona, F. Barceló-Arroyo, and S. Machado, "Implementation and analysis of the aodvv2 routing protocol in arm devices," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, 2021, pp. 1–6.

[9] D. Maltz, J. Broch, and D. Johnson, "Experiences designing and building a multi-hop wireless ad hoc network testbed," 10 2000.

[10] S. Weber, V. Cahill, S. Clarke, and M. Haahr, "Wireless ad hoc network for dublin: A large-scale ad hoc network test-bed," 04 2003.

[11] M. Gharib, A. Owfi, and S. Ghorbani, "Kpsec: Secure end-to-end communications for multi-hop wireless networks," *CoRR*, 2019. [Online]. Available: http://arxiv.org/abs/1911.05126

[12] T. Brown, S. Doshi, S. Jadhav, D. Henkel, and R.-g. Thekkekunnel, "A full scale wireless ad hoc network test bed," 01 2005.

[13] D. Maltz, J. Broch, and D. Johnson, "Quantitative lessons from a full-scale multi-hop wireless ad hoc network testbed," in *2000 IEEE Wireless Communications and Networking Conference. Conference Record (Cat. No.00TH8540)*, vol. 3, 2000, pp. 992–997 vol.3.

[14] E. Nordstrom, P. Gunningberg, and H. Lundgren, "A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks," in *First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*, 2005, pp. 100–109.

[15] P. K. Biswas, C. Serban, A. Poylisher, J. Lee, S.-C. Mau, R. Chadha, C.-Y. J. Chiang, R. Orlando, and K. Jakubowski, "An integrated testbed for virtual ad hoc networks," in *2009 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks Communities and Workshops*, 2009, pp. 1–10.

[16] R. Plestys and R. Zakarevicius, "A testbed for performance evaluation of mobile ad hoc network," in *32nd International Conference on Information Technology Interfaces*, 2010, pp. 155–160.

[17] A. Hussain, A. Khan, A. R. Qaiser, M. M. Akhtar, O. Khalid, and M. F. Khan, "Design and implementation of a testbed for mobile adhoc network protocols," *International Journal of Wireless Communications and Mobile Computing*, vol. 2, no. 4, pp. 42–51, 2014. [Online]. Available: https://doi.org/10.11648/j.wcmc.20140204.11

[18] N. Tabrizi, "An ad hoc networking testbed using software-defined radios," Master's thesis, Middle East Technical University, 2019.

[19] A. Karygiannis and E. Antonakakis, "mlab: An ad hoc network test bed," in *CCNC 2006. 2006 3rd IEEE Consumer Communications and Networking Conference, 2006.*, vol. 2, 2006, pp. 1312–1313.

[20] J. K. Desai, "Testbed implementation of multihop wireless ad-hoc networks," 2019.

[21] E. Nordström, "aodv-uu," https://github.com/erimatnor/aodv-uu, 2011.