12-2023

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

Gage Gailbreath

Andre Koka

Mohammed Gharib

Fatemeh Afghah

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

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

***Abstract*—In communication systems, a Mobile Ad-Hoc Network (MANET) is a communication topology that has no central infrastructure, in contrast to more common network topologies such as Wi-fi or cellular towers. MANETs are of particular interest in the field as Internet-of-Things (IOT) applications and the push towards 6th-generation (6G) communications continues. MANET networks provide communication access to all nodes within the network using peer-to-peer communications, requiring extensive maintenance and updating of routes within the network as nodes move around. Routing protocols must be designed and used for these networks, and are typically complex algorithms that are difficult to implement on hardware. To combat this, this work presents a MANET testbed, designed to provide users with an Application Programming Interface (API) that separates routing protocol implementation from operating system functionality. To verify the testbed, this work also presents an implementation of Ad-Hoc On-Demand Distance Vector Routing (AODV) that uses the provided API functions. By comparing simulation results from Network Simulator 3 (NS3), a physical implementation, and a physical implementation that uses firewall capabilities to form the network, a full evaluation of AODV and the MANET testbed is performed.**

***Index Terms*—MANET networks, AODV, routing protocol, wireless communications, testbed**

I. Introduction

In the field of wireless communications, different network topologies and communication schemes exist for different applications. Many common communication systems use a centralized topology, where there is a single node that facilitates the connection to other nodes in the network. Cellular networks or Wi-fi access points are two prevalent examples of a centralized topology. However, another topology that receives considerably less attention is the decentralized network, which has no underlying infrastructure of any kind. The differences between these networks is shown in Figure 1. The centralized network contains a clear center point, while users in the decentralized network must communicate with other users to form the network.

One type of decentralized network is known as a Mobile Ad-Hoc Network (MANET), where nodes have complete freedom to move in 3D space, communicating to all other nodes in a peer-to-peer fashion. These MANET networks allow nodes to communicate with each other by passing messages through other nodes in the network, which requires the MANET network (and therefore each node within the network) to maintain a database of routes to other nodes. These routes must update as nodes change positions relative to one another, requiring constant network monitoring and complicated routing protocols that must account for changes in real time.

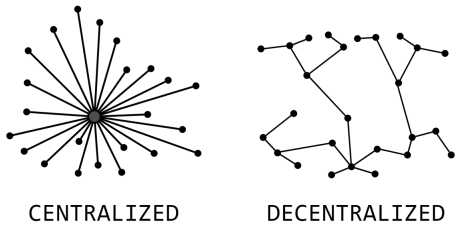

Fig. 1. Centralized vs. Decentralized Network Topologies

The complicated algorithms used for MANET routing protocols are difficult to implement onto a physical system, as modifications to the networking stack of the node's underlying operating system is required to form routes. Different operating systems, operating system updates, and signal propagation characteristics make physical implementations of MANET networks uncommon in wireless communication research. To combat this, this paper presents a MANET testbed that has been developed for further physical implementations of MANET networks. The testbed provides a simple Application Programming Interface (API) that interfaces with the Linux operating system entirely from user-space, using the built-in Linux modules Netfilter and Netlink, as well as Linux UDP sockets and the command-line utility iptables. By doing this, the testbed can be used to implement any experimental routing protocol, allowing researchers to test new protocols on physical systems and move beyond the realm of simulation.

This API relies heavily on the aforementioned Linux modules to send, capture, queue, and filter packets, as defined by the routing protocol. This requires modification of Linux routing tables and communication between user-space and kernel-space using Netlink. This paper presents the design of the MANET testbed, including the API functions and their capabilities. As early verification of the MANET testbed, this work also considers an implementation of Ad-Hoc On-

Demand Distance Vector (AODV) routing, developed in parallel with the MANET testbed. AODV is a complex MANET routing protocol that relies heavily on passing control-plane messages throughout the MANET network, as defined by RFC 3561. While many other MANET routing protocols have been developed that surpass the performance of AODV, AODV is still considered a benchmark route discovery algorithm. Therefore, AODV is used to evaluate the functionality of the MANET testbed. While more extensive testing is needed in this area, qualitative results indicate that the testbed is operational, with a successful implementation of AODV using the API functions. Further testing will be performed to evaluate the AODV environment for the Key Performance Metrics (KPI) throughput and network traffic.

II. Related Works

Now that 5th-generation (5G) communication systems have become common practice for applications such as cellular networks, the research community has turned its focus towards defining what 6th-generation (6G) communication looks like. A prevalent vision for 6G communication systems of the future is the Internet-of-Things (IoT) infrastructure, in which small electronics are used to connect different objects together, creating vast networks of interconnected objects. in [1], Zhou et al. performs an exhaustive study of key technologies that are required to bring 6G IoT networks into reality. They conclude that, with so many small devices communicating at all times, using centralized communication infrastructure becomes impractical for a multitude of small, connected devices, making 6G communications a good use-case of MANET networks [1].

In [2], Inzillo and Garompolo investigate efficient routing protocols for 6G MANET networks by applying a deep learning approach to establish routes. They propose the Deep Learning Clustering Beamforming Massive MIMO Routing Protocol (DLCB) as a new novel routing protocol, concluding that this protocol achieves better simulated results than other routing protocols. [2] serves as another example of general research interest in MANET networks for 6g communication systems. Figure 2 illustrates the evolution of wireless communication systems and helps illustrate the ad-hoc nature of 6G communications. MANETs are expected to be a good option for IoT applications, but still lack effective implementation and testing methods.

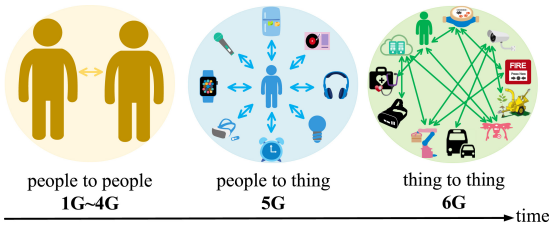

Fig. 2. Wireless Communication Evolution [3]

From here, we move to related works that investigate MANET routing protocols, especially their physical implementations. Erik Nordström is the original creator of AODV-UU, which is the most popular physical implementation of AODV for Linux devices. In [4], he provides the source code and implementation details for AODV-UU, which were last updated in 2011, and have served as a routing protocol benchmark for MANET networks ever since [4].

In [5], Sudharsan et al. extend AODV-UU functionality from Linux kernel 2.6 to Linux kernel 3.8. They update variable names and resolve Qualcomm driver errors, concluding that AODV-UU is working successfully on Linux kernel 3.8. In [6], Jung et al. perform another update of AODV-UU, citing that it is the most stable, open source implementation of AODV. In particular, they implement a custom queue structure to queue packets in absence of a tool like Netfilter Queue, which was not yet included in the Linux kernel. After conducting a simple MANET scenario, [6] concludes with a succesful implementation of AODV-UU on Linux kernel 4.15. There are also many modifications to AODV, some of which have been physically implemented. Zola et al. implements AODVv2, which is a well-known optimization of AODV, on ARM-based devices in [7]. They found that their implementation of AODVv2 was fully operational and intend to provide their code to the research community.

Works featuring the development of a MANET testbed are less common. Staub, Ott, and Braun present an implementation of AODV Mulitpath (AODVM) inside a building on Linux devices in [8]. They evaluate their implementation using an in-house MANET testbed, concluding that AODVM often exacerbates poor route choices and requires the use of multi-channel communication to avoid this weakness. In [9], Lundgren et al. create a large-scale testbed for MANET protocol evaluation, with experiments of up to 37 nodes. After running tests in routes of up to 8 hops, they conclude that the testbed succesfully runs reproducible tests for AODV, TORA, and OLSR, all of which are MANET routing protocols. Finally, Huang et al. perform a comprehensive survey of several large-scale Software Defined Networking (SDN) testbeds in [10]. After providing an overview of SDN testbed characteristics, they conclude that SDN testbeds still require extensive development to properly evaluate routing protocols in a reproducible manner.

A. *Contribution*

Past literature usually focuses on simulation, or presents routing protocol implementations and testbeds that have since become inoperable, especially due to the old age of such publications. In addition, many implementations of AODV and other MANET networks use firewalls and packet dropping rules to manually restrict the connection of nodes in their testing scenarios, which is typically considered a limitation of the physical implementation. The contribution of this paper is the development of a modular, MANET testbed that is independent of both the routing protocol under test, as well as the underlying Linux operating system. In addition, this paper discusses a new implementation of AODV using the testbed, which is usable for actual node mobility tests, both with and without manual firewall creation. Finally, this work

lays the groundwork for MANET protocol comparison using Network Simulator 3 (NS3) and between other MANET routing protocols.

III. Design and Implementation

Design of the MANET testbed was influenced by several factors. Inspiration was drawn from [4] and Erik Nördstrom's original implementation of AODV-UU. In addition, this testbed is the first MANET testbed to be developed on Linux kernel 5.14 or newer, allowing for the use of Linux kernel modules that were previously not included in the Linux kernel by default.

A. User-space and Kernel-space

The MANET testbed is implemented entirely in user-space. There are several advantages to this design decision.

- User-space development allows for the use of common C libraries and network programming.
- User-space code is significantly easier to debug than kernel development. This debugging ability was essential due to lack of experience with kernel or Linux networking development.
- User-space allows for better portability and the abstraction of the routing protocol from the operating system it is running on. This was an important design goal for this testbed.
- User-space development allows for deployment of code updates without needing to recompile the Linux kernel on multiple nodes.

However, developing in user-space has drawbacks as well. The most notable disadvantage is the need to queue network traffic, transfer packets to user-space, then wait for a verdict to be issued for the packet. This manual packet handling is significantly less efficient than the operating system's packet handling capabilities. This leads to a serious limitation of the testbed: *lowered or incorrect performance due to packet queues reaching their maximum capacity.*

B. 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 I 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 3.

TABLE I
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.

C. Provided Functionality

To use the MANET testbed, the user is provided with a simple API consisting of 9 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 9 API functions. The MANET testbed is designed to abstract the operating system details away from the user. Therefore, each API function implements some kind of operating system 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, formating, and sending a Netlink message as part of the routing protocol implementation code. The 9 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 it allows the routing protocol to be detached from how it interacts with the hardware. This abstraction allows upgrading and optimization

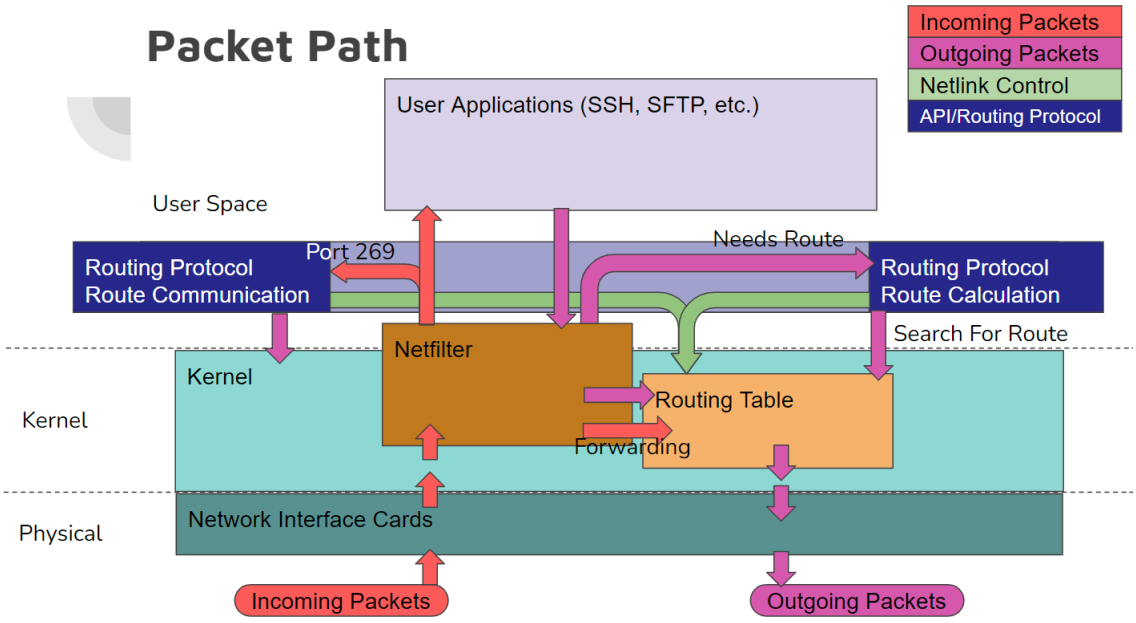

Fig. 3. MANET Testbed Architecture

independent of each other. A similar API can then be made in kernel space or in 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. And researchers for the protocols can compare multiple protocols written using the same API without having to re-implement it themselves, unifying the comparisons.

D. Implementation Verification: AODV

AODV is a standard routing protocol that has been used as a benchmark to evaluate other MANET routing protocols. Since the inception of AODV, several modifications of AODV, as well as completely original route discovery algorithms, have also surfaced. Despite this, AODV was chosen as the baseline protocol to inform testbed development. 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 this order:

1) Check if a current, active route exists for this packet. If not, proceed with route discovery.
2) Broadcast a Route Request packet (RREQ).
3) Broadcast with increasing Time-To-Live (TTL) until either maximum attempts are reached or a Route Reply (RREP) is returned.
4) Drop or send the packet depending on result of route discovery.

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.

Since thorough explanations of AODV are present in many papers, the rest of this section will instead address the features absent and design choices made to feasibly implement AODV. First, to store a local routing table, a static array was used. Knowing the physical limitations of the project, this would maximize performance and minimize development time, but this array data structure was still developed in a modular way such that it can easily be swapped out. Second, ”Hello” messages were chosen as a way of maintaining link connectivity. The RFC for AODV (RFC 3561) states that these messages are a way of ensuring connectivity, but they have the drawback of allowing up to two seconds (depending on the ”Hello” message interval) where potential messages may be lost. AODV has other methods of checking link connectivity, but this implementation assumes that the MANET network is not a highly-dynamic environment, meaning that ”Hello” messages will suffice. The features missing from AODV are as follows:

1) Unidirectional Links: Unidirectional link accommodation was not implemented for this project. The major limitation was time, but skeleton code was written to allow easy inclusion in the future.

2) Local Repair: As the focus of this development was on the MANET testbed and its verification instead of the protocol itself, local repair was not included as it would require a more

significant topology and testing scenario to be relevant.

3) Actions After Reboot: Similar to local repair, as scenarios and focus of this paper were more limited, implementation of this feature was not required. If this feature was pursued, the focus of the project would shift from the physical implementation to the testing of AODV itself.

4) More than One Interface/Aggregated Networks: Similar to the previous features, implementation of these aspects would be an assessment of the routing protocol and not the testbed itself. The MANET testbed allows for multiple interfaces and aggregated networks, but it is up to the protocol and protocol designer to account for them.

Outside of these features, the rest of the operation of AODV was implemented using III-C. This provides a protocol to assess the testbed and the performance of its various functions.

IV. Results

With the MANET testbed and AODV implementation completed, testing was performed by simply running the AODV implementation, which makes use of all 9 API functions. Manual inspection of behavior shows that AODV is working correctly. Figure 4 shows a representation of the AODV test scenario that was used. This is a 2-hop connection, in which a source and destination node must communicate with each other using a single intermediate node. Many such test scenarios were performed by placing the nodes in different parts of the building and powering them using portable power banks.

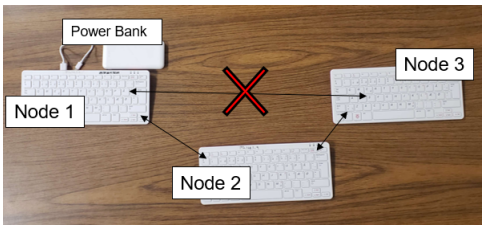

Fig. 4. Representation of 2-Hop Test Setup

However, while results of the AODV implementation indicate that the MANET testbed is fully operational, there are not yet many quantitative results that can be used to categorize the performance of AODV on the MANET testbed. Qualitatively, the AODV implementation is successful, with route discovery capabilities according to the AODV protocol working appropriately. Route requests are properly broadcast throughout the MANET network, responded appropriately, and the corresponding routes are installed in each node's routing tables as needed. Since this implementation of AODV relies on all 9 functions of the MANET testbed API, the MANET testbed is also fully functional.

During the course of testing, the network and the nodes within it would occasionally exhibit unexpected behavior. This includes nodes disconnecting randomly, nodes connecting from too far away, and failure to send packets despite the successful establishment of a route. Such behavior can be attributed to the testing setup and environment. This is a wireless communication system, which are heavily affected by physical obstructions. Walls, people, and other moving obstacles in the network result in erratic signal propagation characteristics. In addition, the portable power banks only supply power for a limited time before needing to be recharged, resulting in situations where a node is still powered, but not with enough voltage to guarantee proper operation. Despite these issues, several tests resulted in proper AODV operation, which was used to succesfully send files and maintained routes even when mobility was introduced to the system. Therefore, the MANET testbed and the AODV implementation that makes use of it are considered a success.

V. Future Work

This work can be extending in several ways, but the most important is to address the weaknesses of the implemented MANET testbed:

- **Queue Status** - Implement the ability to check when packet queues are full. This could potentially be accomplished using an `nftables TRACE` rule to check when packets are no longer being queued, but remains a difficult problem with no clear solution.
- **Testing Capabilities** - Testing capabilities through the testbed are limited. Currently, timing metrics are included in the AODV implementation, indicating that the testbed is not completely abstracted from the routing protocol.

Other than addressing limitations, future work includes implementing more MANET routing protocols with the testbed, evaluating complex mobility scenarios, and performing comparisons between routing protocols and their simulated versions, through a tool such as NS3.

VI. Conclusions

This paper presents a MANET testbed, which boasts the ability to implement any MANET routing protocol on Linux devices running kernel 5.14 or newer, while abstracting the operating system details away from the implementer of the routing protocol. This work also uses a custom implementation of AODV to evaluate the testbed. Results show that all API functions are fully operational, but require further development to include testing metrics and overall more robust capabilities. In general, this work illustrates the need for further physical implementations of MANET networks, as implementing and comparing MANET routing protocols still remains a difficult task, with no clear standard in the field of communication systems.

VII. Andre's Discussion of His Work

I joined Gage on this project after my initial projects that I started with the lab began to slow down, leading to a lack of activity for me in the lab. Since then, I have assisted Gage in designing new parts of the project and implementing the MANET testbed based on his original design. Over the summer, I focused on learning Gage's testbed design and performing some other work with the lab. This semester, testbed implementation, data collection, and overall verification of the

system became the focus. This was an ambitious project that required me to think independently and create solutions to extremely specialized problems. I have gained extensive Linux kernel knowledge and improved my code-writing abilities, while learning more about wireless communication systems and becoming an expert on this topic. I have also gained significant experience with writing research papers, writing literature reviews, and presenting my work to others. This MANET testbed has several limitations, but proved to be an invaluable experience and an overall successful implementation of a very difficult task.

VIII. Gage's Discussion of His Work

When I initially started this project last semester, I was the only student working on it and I did not have any background in it. My first step was building a large repository of resources and papers to begin basing my work off of. The initial focus was to just make and evaluate the testbed, AODV had not yet been established. By the end of last semester, I had a full plan of how the testbed can be accomplished in terms of Linux utilities and interactions. I created figures and organized all of my resources so they could be handed off to Andre over the summer. This semester, while Andre finalized the testbed, I investigated AODV and how to implement it. This ended up being a large task as you are essentially turning a high-level 26 page technical specification into fully functioning code. In addition to this, with the testbed not fully completed, the majority of the code was written untested. Despite this, I was able to accomplish the stripped down version described in Section III-D. I then moved onto testing where I first did a code check through looking for segfaults or other erroneous behavior. It was during this time that I found some logical issues with the implementation of the RFC where the behavior was undefined so I had to think of my own solution. The final step I am doing is getting preliminary data and metrics to finalize the project and provide quantifiable results. This is what I am currently working on and am hoping to finish the data acquisition for analysis after this semester. Over the course of this project I learned a lot about routing protocols, the Linux kernel networking stack, software development practices, and general networking knowledge. It made me work with utilities and tools that had not been updated in years and learn how to adapt my solutions and old knowledge to the changes in Linux.

IX. Acknowledgment

This study is supported by the Intelligent Systems and Wireless Networking (IS-WiN) laboratory at Clemson University. Special thanks to Dr. Fatemeh Afghah, Dr. Mohammed Gharib, the Clemson Electrical and Computer Engineering department, and the Clemson University Honors College for making this work possible.

References

[1] Y. Zhou, L. Liu, L. Wang, N. Hui, X. Cui, J. Wu, Y. Peng, Y. Qi, and C. Xing, "Service-aware 6g: An intelligent and open network based on the convergence of communication, computing and caching," *Digital Communications and Networks*, vol. 6, no. 3, pp. 253–260, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864820300237

[2] V. Inzillo and D. Garompolo, "A deep learning clustering beamforming approach for future 6g mobile ad hoc networks," 07 2023.

[3] Mar 2023. [Online]. Available: https://www.geeksforgeeks.org/introduction-of-mobile-ad-hoc-network-manet/

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

[5] V. Kasula, "Implementation of aodv-uu on linux kernel version 3.8," 04 2014.

[6] S. Jung, B.-S. Kim, K.-I. Kim, B. Roh, and J.-H. Ham, "Implementation of aodv-uu on linux 4.15 kernel," in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, 2019, pp. 160–161.

[7] 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.

[8] T. Staub, S. Ott, and T. Braun, "Experimental evaluation of multi-path routing in a wireless mesh network inside a building," *ECEASST*, vol. 17, 01 2009.

[9] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordstrom, and C. Tschudin, "A large-scale testbed for reproducible ad hoc protocol evaluations," in *2002 IEEE Wireless Communications and Networking Conference Record. WCNC 2002 (Cat. No.02TH8609)*, vol. 1, 2002, pp. 412–418 vol.1.

[10] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang, and Y. Liu, "A survey on large-scale software defined networking (sdn) testbeds: Approaches and challenges," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 891–917, 2017.