AODVv2 in ns-3: Implementation and Early Results

Francesco Todino
Dept of Information Engineering
University of Florence
Florence, Italy
francesco.todino@edu.unifi.it

Tommaso Pecorella
Dept of Information Engineering
University of Florence
Florence, Italy
tommaso.pecorella@unifi.it

Charles Perkins
Blue Meadow Networks
Saratoga, CA, USA
charliep@computer.org

Flavio Esposito
Saint Louis University
Saint Louis, MO, USA
flavio.esposito@slu.edu

Abstract

The increasing complexity and scale of modern network environments, particularly with the advent of the Internet of Things (IoT), demand flexible and adaptable routing protocols. Our study delves into the first known ns-3 implementation of Ad Hoc On-Demand Distance Vector version 2 (AODVv2), offering researchers a valuable platform to explore its capabilities. This implementation facilitates rigorous testing and comparative analysis under diverse network conditions, leading to new insights on the protocol's performance and reliability. Key findings from our preliminary evaluations indicate that AODVv2's overhead is higher than that of legacy AODV, but its route maintenance is more robust in dynamic topologies. These results accelerate innovation in mobile ad hoc networking and IoT domains, encouraging the development of more adaptive and resilient networks. Furthermore, our work offers empirical results to inform future standardization efforts for AODVv2.

CCS Concepts

• **Networks** → **Network simulations**; **Ad hoc networks**; **Routing protocols**.

Keywords

AODVv2, IPv6, MANET, routing, multi-metric

ACM Reference Format:
Francesco Todino, Tommaso Pecorella, Charles Perkins, and Flavio Esposito. 2025. AODVv2 in ns-3: Implementation and Early Results. In *2025 International Conference on ns-3 (ICNS3 2025), August 19–21, 2025, Osaka, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3747204.3747219

1 Introduction

The rapid expansion of modern network environments, driven by the proliferation of IoT devices, has created a pressing need for routing protocols that can efficiently manage dynamic and unpredictable network conditions. Traditional routing protocols often fall short in these scenarios, necessitating the development of more flexible and adaptable solutions. Ad Hoc On-Demand Distance Vector version 2 (AODVv2) [12] is a promising reactive routing protocol designed to address these challenges. Despite its potential, AODVv2's standardization and adoption has been slow, mostly due to the absence of a readily available and open-source implementation that can be rigorously tested and validated in realistic scenarios.

This paper addresses this critical gap by presenting the first *comprehensive* AODVv2 implementation in the widely used ns-3 simulator [10]. The ns-3 simulator is known for its ability to model and simulate complex network environments, making it an ideal platform for exploring the capabilities of AODVv2. By integrating AODVv2 into ns-3, we provide researchers and developers with a powerful toolset to analyze the protocol's performance under diverse conditions.

Our contributions are threefold:

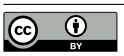
- We detail the design choices and architectural structure of the AODVv2 implementation, including its IP-agnostic approach and metric extensibility.
- We evaluate the performance of AODVv2 against AODV, demonstrating where AODVv2 excels and identifying areas needing optimization, such as packet overhead.
- We offer this implementation as an open-source resource for the research community, aiming to foster deeper investigations into AODVv2 for IoT and MANET use cases.

In the following sections, we discuss the state of the art in AODV and AODVv2 research, summarize the key design differences between the two protocols, and present our ns-3 integration strategy. We then evaluate the performance of our implementation through an extensive set of simulations, highlighting the protocol's strengths and identifying opportunities for further improvement. Finally, we discuss the path forward for refining AODVv2 and integrating it with emerging network infrastructures.

By providing a robust and flexible foundation for AODVv2 research, we hope to accelerate innovation and encourage the creation of more adaptive and resilient network infrastructures in the evolving IoT landscape. We also hope to encourage the publication of AODVv2 as a Standards Track document in the IETF.

2 Related Work

Ad Hoc On-Demand Distance Vector (AODV) routing protocol [11], initially standardized in RFC 3561 [5], has been extensively studied and implemented since its publication in 2003. The protocol's effectiveness in mobile ad hoc networks (MANETs) is reflected in



This work is licensed under a Creative Commons Attribution 4.0 International License.
ICNS3 2025, Osaka, Japan
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1517-4/25/08
https://doi.org/10.1145/3747204.3747219

the substantial body of research, with almost 30,000 citations to the original RFC. Based on the popularity of Experimental Protocols AODV and DSR [7], the IETF [manet] working group undertook the task of unifying the two on-demand protocols into a Proposed Standard. This unified protocol would also address AODV's limitations, for example its IPv4-only design and inability to accommodate multiple metrics for route selection.

AODVv2's development in the IETF MANET working group began in 2005, progressing through multiple Internet Drafts but not achieving RFC status.

Current research on AODVv2 is limited by the lack of accessible and standardized implementations. For instance, the only known implementation is https://github.com/Lotterleben/RIOT-AODVv2 inside RIOT OS but is dated (since there are more than 10 reviews of the draft since their last repository update) and it is not so easy to use to compare with other protocols. A bibliometric analysis reveals that while AODV appears in thousands of research papers, AODVv2 is referenced in fewer than 300 publications, mostly focusing on theoretical analyses rather than practical implementations. Some implementations have been developed, but all of them are not tailored to be able to compare and analyze the behavior of the last version of AODVv2 with other protocols [8, 14].

This lack of suitable and up-to-date implementations makes it difficult for researchers to perform fair and reproducible evaluations of AODVv2 in realistic settings. This disparity is particularly notable in simulation environments, where AODV models are readily available in simulation platforms like ns-3, while AODVv2 implementations remain unavailable. This implementation gap severely restricts comparative studies, protocol validation, and performance analysis across diverse network scenarios.

Our contribution of an AODVv2 implementation for ns-3 addresses this critical research gap, providing a standardized platform for protocol evaluation and refinement. This implementation is not only valuable for advancing the research on AODVv2 but also essential for generating empirical evidence that could help the standardization process within the IETF or other standardization bodies.

3 AODV and AODVv2

AODV (Ad Hoc On-Demand Distance Vector) is a well-established reactive routing protocol designed for MANETs. It works by discovering routes on-demand, which helps to minimize the overhead associated with maintaining a complete routing table. AODV is widely cited and has been the subject of extensive research due to its simplicity and effectiveness in dynamic network environments. AODVv2, on the other hand, is an evolution of AODV that addresses several limitations of its predecessor. One of the most significant differences is that AODVv2 is designed to be agnostic to IP versions, supporting both IPv4 and IPv6, whereas AODV is limited to IPv4. This makes AODVv2 more versatile and suitable for modern network environments that increasingly rely on IPv6.

Another key difference is in the packets: AODV uses a custom packet format, while AODVv2 adopts the Generalized MANET Packet/Message Format [1], which is also used by OLSRv2 [6]. The use of a unified packet format facilitates protocol extensions (e.g., new metrics), and simplifies the development and testing, because the actual packet format can be developed once and reused across different protocols.

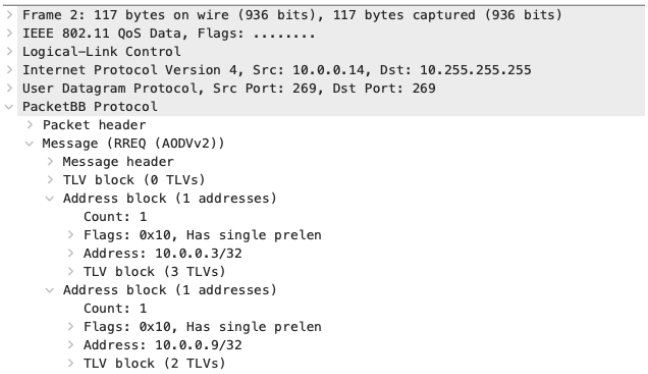


Figure 1: An RREQ Packet Dissected by Wireshark

AODVv2's design and standard packet format also accommodates multiple routing metrics beyond the traditional hop count used in AODV. The definition of new metrics is particularly important in IoT and MANET contexts, and is a key point in modern routing protocols. As an example, RPL [2] allows multi-metric compositions, also in the form of constraints.

Additionally, AODVv2 includes mechanisms to confirm bidirectional links by requesting acknowledgments for each RREP message, which helps to avoid routes through unidirectional links and improves overall network reliability.

Furthermore, AODVv2 reduces the amount of routing information that must be maintained, making it more suitable for resource-constrained devices common in IoT deployments. In fact, AODVv2 intentionally removes certain features present in AODV to create a more lightweight protocol. Specifically, it removes hello messages that were used in AODV for local connectivity management, opting instead other feedback mechanisms when available, e.g., layer2, NHDP [4], etc. Notably, Wi-Fi assures that links operate in both directions, as do other common wireless technologies, so that Hello messages are superfluous.

Additionally, AODVv2 eliminates the expanding ring search technique that AODV uses to gradually increase the search area for routes. These deliberate omissions streamline the protocol's operation, reduce control overhead, and make AODVv2 particularly suitable for networks with constrained devices.

Figure 1 shows a dissection of an RREQ packet in Wireshark, highlighting the TLV structure used in AODVv2.

4 AODVv2 Design Choices

The implementation of AODVv2 for IoT networks required careful consideration of design choices to ensure flexibility, extensibility, and compatibility with future network protocols. Rather than extending the existing AODV codebase, we opted for a complete redesign that would better accommodate the distinct features of AODVv2, particularly its Type-Length-Value (TLV) encoding structure and IP version agnosticism.

A key design decision in our implementation was the adoption of a template-based approach. Given that AODVv2 is IP version agnostic, our goal was to implement it with support for both IPv4 and IPv6. To achieve this, we designed every class and method to be templated, allowing the protocol to be compatible with any IP version. By templating the `Aodvv2RoutingProtocol` class, we created a single codebase that can be instantiated for either `Ipv4RoutingProtocol` or `Ipv6RoutingProtocol` without code duplication.

This design has been inspired by the `nix-vector-routing` model, which has a similar structure. The use of templates simplifies the development and verification of the protocol. However, it also highlights some notable drawbacks.

The first drawback is represented by the use of 'neutral' names, e.g., `IpAddress` instead of `Ipv4Address` or `Ipv6Address`. This type of aliasing is made possible by using `std::conditional_t`, but it might be confusing for someone new to the code.

The second drawback is related to the first: Despite the aliases, conditional blocks are still needed because the interfaces for the different classes are slightly different. As an example, `Ipv6Address` does not have an `IsBroadcast` function. This might extend to entire functions, or to the callbacks definitions, which adds complexity to the solution.

However, the above-mentioned drawbacks are, in our opinion, limited to some code parts (sockets initialization, route input and output, etc.) that do not need to be changed often, and that are easy to test and verify.

On the other hand, using an IP-agnostic representation simplifies the 'logic' part of the standard, i.e., generating and reacting to control messages, which is the most critical part of the protocol.

The AODVv2 implementation consists of several interdependent classes that together form a cohesive routing system. Figure 2 illustrates the relationship between these components through a UML class diagram.

The main class, `Aodvv2RoutingProtocol`, manages the protocol's core functionality, including packet handling, route management, and timer maintenance. This class implements virtual functions such as `RouteOutput` for locally originating packets and `RouteInput` for forwarding received packets.

Supporting classes handle specific aspects of the routing protocol, starting from some of the ones used to represent the protocol sets:

- `RouterClientSet` maintains information about local applications and devices.
- `NeighborSet` tracks neighboring AODVv2 routers with state management.
- `LocalRouteSet` stores routing information with state transitions (*UNCONFIRMED*, *IDLE*, *ACTIVE*, *INVALID*).
- `MulticastMessageSet` prevents redundant processing of multicast/broadcast messages
- `RouteErrorSet` records routing errors to manage error propagation.

AODVv2 defines four primary control packets: Route Request (RREQ), Route Reply (RREP), Route Reply Ack (RREP_Ack), and Route Error (RERR). Each packet type is implemented as a separate class with specialized attributes and methods for encoding and decoding TLV structures.

The packet implementation leverages ns-3's `packetbb` model to handle TLV encoding, with custom functionality for header management through methods like `SetTlvHeader` and `CreateTlvHeader`. As depicted in Figure 3, the control packet classes share similar structures but implement specialized behaviors.

The metric system represents one of the most innovative aspects of our AODVv2 implementation. The AODVv2 draft only requires an hop-count metric, similar to the AODV one. Rather than hard-code the metric, we designed an extensible framework that allows the addition of new metrics without modifying the core protocol code. The `Metric` class, shown in Figure 2, provides the foundation for this extensibility.

The class defines three essential methods that any metric must implement:

- `LinkCost`, which calculates the cost of a single link;
- `RouteCost`, which determines the aggregate cost of a complete route; and
- `LoopFree`, which ensures that route updates do not introduce routing loops.

This design allows developers to create custom metrics tailored to specific network requirements. For instance, one could implement energy-aware metrics for battery-constrained IoT devices, latency-optimized metrics for time-sensitive applications, bandwidth-occupancy metrics, or other types of metrics.

The extensibility of our metric framework is particularly valuable in IoT environments, where network conditions and requirements can vary dramatically between deployments. A developer wishing to add a new metric needs only to create a class that inherits from the `Metric` base class and implements the required methods. The flexibility in extending the metrics has been used in [13].

The metric structure is shown in Listing 1. It is worth noticing the use of lambda functions to define the three functions, which allows to define a metric in the user's code without modifying the model. However, it is also necessary to highlight a couple of choices, mainly due to rapid development. First and foremost, the memory allocation in the `LinkCost` and `RouteCost` functions could be avoided. The second is the use of a `MetricNode` class, which is needed to access the node's parameters (e.g., the node energy), and that will be removed as well, as it is unnecessary. While we do not foresee major changes in the architecture, the code will be improved before intgrating it with the main ns-3 tree.

The template-based implementation, modular class structure, and extensible metric framework collectively form a robust foundation for AODVv2 in ns-3. This architecture not only fulfills the current requirements of the protocol but also anticipates future extensions and adaptations to evolving network environments.

5 AODVv2 Tests and Performance Evaluation

We have integrated a series of comparative tests involving AODV to enable a comprehensive comparison with our implementation of AODVv2. The goal of this phase of experimentation is to evaluate and benchmark each protocol performance under identical network conditions, highlighting their respective strengths and weaknesses, and to provide a baseline against which our proposed solution can be measured.

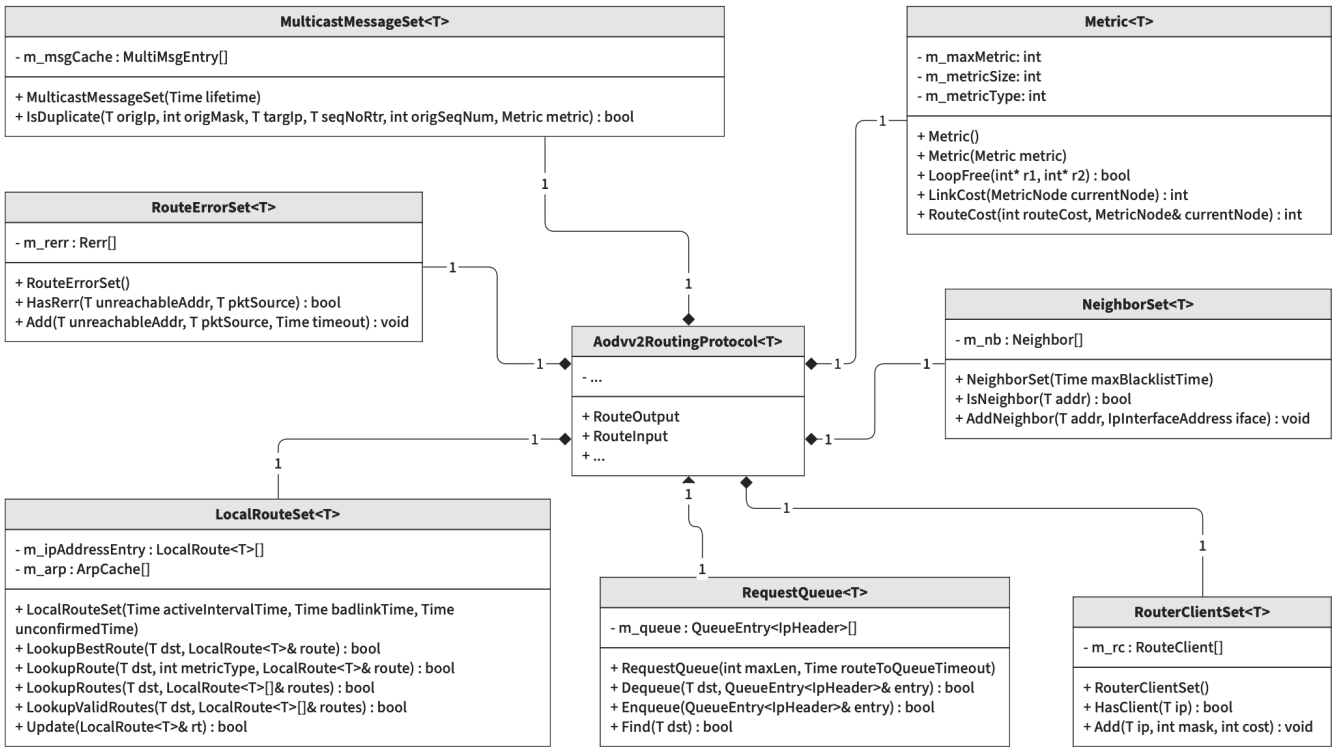


Figure 2: AODVv2 Protocol Architecture UML Diagram

Listing 1: Hop Count Metric Code Structure

```
Metric()
{
    m_metricType = AODVV2_METRIC_HOP;
    m_metricSize = 1;
    m_maxMetric = 255;
    m_linkCost = [](const MetricNode&) { return new uint8_t[1]{1}; };
    m_routeCost = [](const uint8_t* routeCost, const MetricNode& currentNode) {
        uint8_t* newCost = new uint8_t[1];
        newCost[0] = static_cast<uint8_t>(routeCost[0] + 1);
        return newCost;
    };
    m_loopFree = [](const uint8_t* r1, const uint8_t* r2) {
        return r1[0] <= r2[0];
    };
}
```

We performed two sets of experiments. In the first, we used the `manet-routing-compare.cc` script. In the simulation there is a path disconnection around second 150, and a following path recomputation.

In the second set of experiments, the nodes have a fixed random position in a box measuring 150m by 150m. In this case, we did increase the number of nodes in the simulation, thus increasing the density of the network, and we repeated each experiment 50 times, varying the simulation seed (hence, both the nodes position and source/destination pairs). The simulations last 10 seconds, with the nodes beginning to exchange ICMP Echo packets as soon as they can, and continuing until the end of the simulation, with a rate of 1 packet per second. This scenario setup has been deliberately chosen to highlight traffic interference between the nodes.

In both cases, we used the `Ping` application, where in the first experiment the node pairs have not been changed with respect to the ns-3 example, and in the second case we select randomly half the nodes in the simulation, and we let them communicate with a

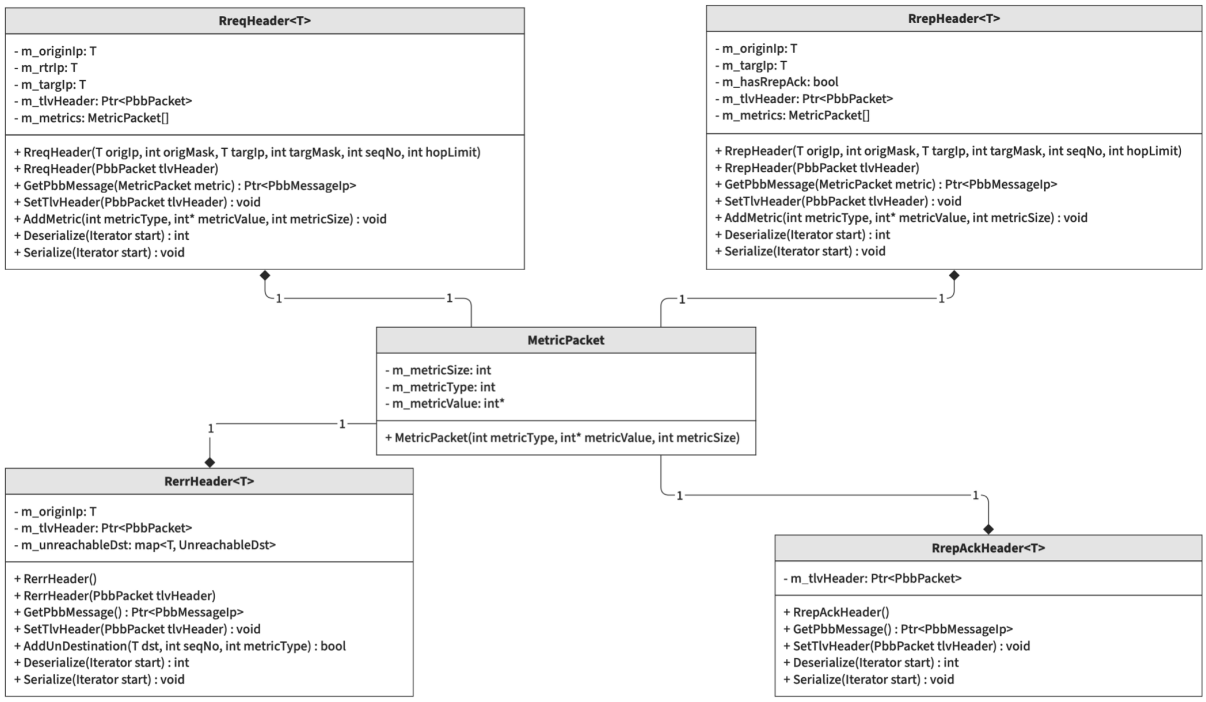


Figure 3: Control Packet Classes UML Diagram

different node with at least one hop in between (i.e., avoiding pings between immediate neighbors).

For the sake of simplicity, and without loss of generalization, the simulations are using IPv4.

Since AODVv2 is still a draft, results are in preliminary form, and enhancements are possible. We expect to see that AODV will perform better in many of the results due to its smaller packet size.

5.1 Packets Overhead

As illustrated in Table 1, each protocol produces a different number of control messages with varying sizes, reflecting the anticipated behavior of the protocols. AODVv2 uses half the number of RREQ messages compared to AODV to establish routes, but the size of these packets is nearly double due to the standardized message format and the lack of packet compression. Furthermore, the significantly higher number of RREP and RREP_Ack messages in AODVv2 compared to AODV is because AODVv2 is designed to verify bidirectional links to avoid the creation of asymmetric routes.

As discussed later, the overhead difference has direct effects on the protocol performances, and represents one of the critical points to consider in the implementation and, possibly, future protocol optimizations.

Table 1: Protocols Overhead (Not Including IP Header)

Type	Avg Count [#]	Average [Bytes]	Min [Bytes]	Max [Bytes]
AODV				
RREQ	110.78	24.0	24	24
RREP	194.63	20.0	20	20
RERR	2.00	12.0	12	12
RREP_ACK	1.74	2.0	2	2
AODVv2				
RREQ	58.98	49.4	46	51
RREP	306.83	53.9	46	54
RERR	3.94	60.7	40	120
RREP_ACK	306.30	11.0	11	11

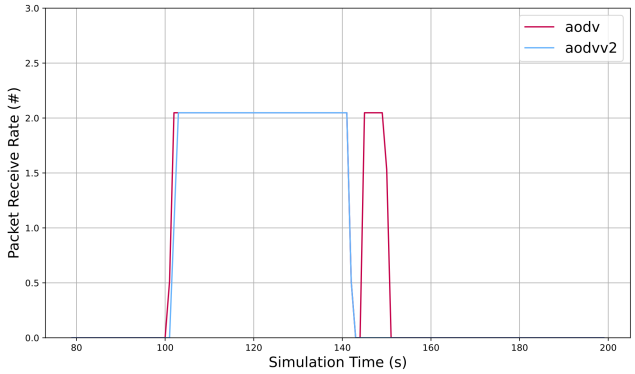


Figure 4: Packet Receive Rate Comparison without External Mechanisms. As we can See, after 150 Seconds, Protocols Stop Sending Packets as they Are not Updated and Routes Have not yet Expired

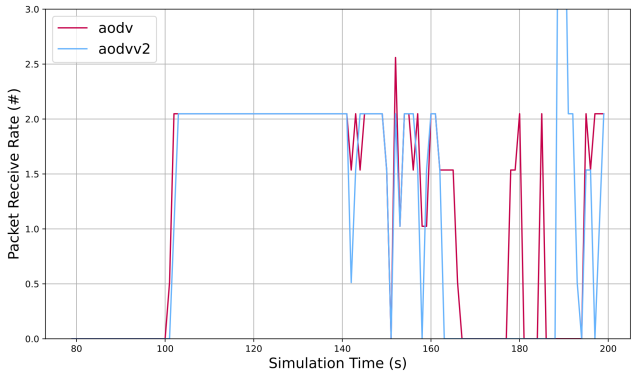


Figure 5: Packet Receive Rate Comparison with External Mechanisms. Here, with the WiFi Module Update, we Keep Sending Packets after 150 Seconds

5.2 Packet Receive Rate

In this comparison, we analyze the importance of external mechanisms for protocols to keep routes updated.

In the `manet-routing-compare.cc` script, there is a network disruption around second 150. As we can see from Figure 4, the curve of AODV shows a stable trend with almost the same performance as the other protocol. Starting from 150 seconds, both protocols stop sending packets, because nodes are still trying to use the old routes that are no longer valid but not expired.

Figure 5 adds the WiFi module as an external update mechanism, and the AODV curve shows improved performance, being able to send packages even after the drop analyzed before. AODVv2 seems to have similar behavior, but with a lower packet receive rate, which can be explained by the protocol choice of sacrificing some performance to reduce the service packets sent while respecting the protocol specifications and timeouts.

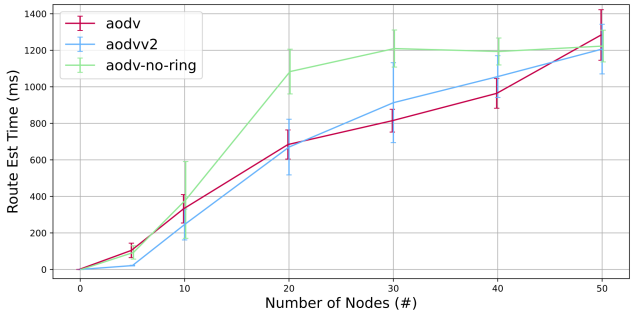


Figure 6: First RTT Experienced by each Node Pair. AODV and AODVv2 Have Similar Results while AODV without Expanding Ring Works Worse

5.3 Round-Trip Time

In Figure 6, we compared the time required by the protocol to establish a route by measuring the time to receive the first Echo reply.

It is clear how in a moderately dense network (between 5 and 20 nodes), AODVv2 outperforms AODV, and the lack of ring expansion makes things worse for AODV. As a matter of fact, the scenario area is small enough to allow an average distance between the source and destination of 2 or 3 hops. When the network becomes more dense, AODV benefits from the ring expansion mechanism, which allows searching for the destination by progressively increasing the search radius. Disabling the ring expansion mechanism creates a larger overhead, as shown by the difference between the two AODV curves.

It is very interesting to observe that while AODVv2 does not have any ring expansion search, the performance is very good, and always superior to AODV without ring expansion.

In Figure 7 we measured the average round trip time of the various protocols. As the three protocols are all selecting the minimum hop path, we would expect similar, if not identical results. Nevertheless, we see that AODVv2 performs slighly worse than AODV. This result can be explained by the overhead of the control packets. As noted in section 5.1, AODVv2 packets are much larger than the AODV ones, creating a background noise that affects the ping performance.

Since both AODV and AODVv2 are selecting the minimum hop path, we expect that the end-to-end delay becomes similar once the path are all established, i.e., when the network is in steady-state. However, in this experiment we kept the transient phase going on in order to evaluate the effects of the larger overhead on the network. The results are evident: the size of the packets does negatively affect the system.

6 Path Forward

While our AODVv2 implementation represents a significant step forward for routing protocol research in ns-3, several limitations and opportunities for enhancement remain. Addressing these issues will be the focus of our future work.

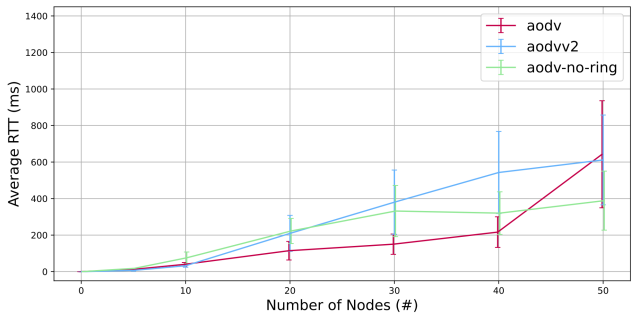


Figure 7: Average RTT over the Entire Simulation. As the First RTT Analysis, AODV Performance without Expanding Ring is Worse than AODV but Congestion still Degrades AODVv2 Performance

Our implementation currently lacks support for interaction with external networks, i.e., AODVv2 cannot be used as a "transit" network, and setting a default gateway in the nodes is cumbersome. This limitation prevents researchers from testing hybrid scenarios where simulated AODVv2 networks communicate with real-world infrastructure, which would be particularly valuable for validating the protocol's behavior in IoT gateway applications.

As demonstrated in our performance evaluation, the standardized packet format used by AODVv2 results in larger control messages compared to AODV. Our implementation currently lacks efficient address compression mechanisms that could significantly reduce this overhead. Future work may explore compression approaches (e.g., [3, 9]) and 6LoWPAN-inspired techniques to decrease message sizes, which could help mitigate the performance gap between AODV and AODVv2 performance.

While our testing is not strongly affected by this, we have to note that there is a rather strong limitation in the ns-3 `packetbb` model: it does not support Address blocks TLV packing. While this does not completely invalidate simulations with IPv4 (where the number of zeroes in an address is limited), it creates issues when multiple addresses are included in a message and when IPv6 is used. As an example, transmitting 3 addresses (e.g., 169.254.0.1, 169.254.0.3, and 169.254.0.42) should require only 5 bytes, and not 24.

We emphasize that the `packetbb` optimizations are necessary, but are transparent to AODVv2. In other terms, the choice of how to pack/compress the TLVs should be handled by the `packetbb` model, and not by the AODVv2 protocol. The two types of data compression/optimization (i.e., in AODVv2 and packetbb) are not mutually exclusive, and their efficiency should be evaluated together.

This consideration further validates the use of `packetbb` as a 'common' packet format for AODVv2, OLSRv2, and other MANET protocols: while the packet is normally larger, all the protocols can benefit from the optimizations in the packet format model.

While we have conducted preliminary comparative testing with AODV, our implementation would benefit from more extensive validation across a broader range of network topologies, mobility patterns, and traffic profiles. The current test suite does not fully explore edge cases or extreme network conditions that could reveal additional optimization opportunities or potential protocol weaknesses.

7 Reproducing the Experiments

The model presented in the present work can be found on the `aodvv2-icns3-25` branch of the author's GitLab repository, at https://gitlab.com/tod97/ns-3-dev/-/tree/aodvv2-icns3-25.

The branch will be kept stable for the seek of reproducibility; the development will continue in different branches, with the aim to merge the model in ns-3 when the model will be fully validated and, possibly, when AODVv2 will become an RFC.

The repository contains instructions to reproduce the results in the `README.md` file.

The authors will keep an archive of the software in case the repository fails.

8 Conclusions

This paper presents the first implementation of AODVv2 in ns-3, addressing a significant gap in the research and development of adaptive routing protocols for modern network environments. Our implementation provides researchers with a valuable platform to explore, test, and refine this promising protocol, particularly for IPv6-based IoT and MANET deployments where traditional routing solutions are inadequate. Our comparative analysis with AODV reveals both expected trade-offs and opportunities for optimization.

While AODVv2 demonstrates slightly higher control overhead and latency due to its standardized packet format and bidirectional link verification mechanism, these features ultimately contribute to greater protocol versatility and reliability in complex network environments. The protocol's IP-agnostic design and support for multiple routing metrics position it as a more flexible solution for heterogeneous networks.

The implementation described in this paper not only serves as a research tool but also has the potential to accelerate the AODVv2 standardization process by facilitating evidence-based refinements. While the code used for this paper is not yet ready for merging into ns-3, the modifications will be minor, mainly consisting in code cleanup, debugging, and eventually changes to make it compliant with the future versions of the draft.

Future work will focus on optimizing packet compression techniques, exploring more sophisticated metrics beyond hop count, and extending the implementation to support external network integration. Additionally, we plan to conduct more extensive performance evaluations across diverse network scenarios to further validate and improve the protocol's effectiveness. We hope to identify those scenarios in which reactive protocols are expected to perform better than proactive protocols.

By contributing this implementation to the ns-3 ecosystem, we aim to foster broader adoption of AODVv2 and stimulate innovation in routing protocols for the increasingly complex and dynamic network environments that characterize modern IoT and ad hoc networking applications.

Acknowledgments

The authors would like to thank Prof. Mohit P. Tahiliani from the National Institute of Technology Karnataka (NITK) and his students, Aniket Singh, Satyam Shukla and Anirudh V. Gubbi for their support in the development of AODVv2 test cases. This work has been supported by NSF award OAC # 2201536.

References

[1] Cédric Adjih, Christopher Dearlove, Justin Dean, and Thomas H. Clausen. 2009. Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format. RFC 5444. doi:10.17487/RFC5444

[2] Roger Alexander, Anders Brandt, JP Vasseur, Jonathan Hui, Kris Pister, Pascal Thubert, P Levis, Rene Struik, Richard Kelsey, and Tim Winter. 2012. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550. doi:10.17487/RFC6550

[3] Carsten Bormann. 2014. 6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). RFC 7400. doi:10.17487/RFC7400

[4] Thomas H. Clausen, Christopher Dearlove, and Justin Dean. 2011. Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP). RFC 6130. doi:10.17487/RFC6130

[5] Samir R. Das, Charles E. Perkins, and Elizabeth M. Belding-Royer. 2003. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561. doi:10.17487/RFC3561

[6] Christopher Dearlove and Thomas H. Clausen. 2014. Optimized Link State Routing Protocol Version 2 (OLSRv2) and MANET Neighborhood Discovery Protocol (NHDP) Extension TLVs. RFC 7188. doi:10.17487/RFC7188

[7] Yih-Chun Hu, Dave A. Maltz, and David B. Johnson. 2007. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728. doi:10.17487/RFC4728

[8] Sergio Machado, Israel Martin-Escalona, Enrica Zola, and Francisco Barcelo-Arroyo. 2021. A User Space Implementation of the AODVv2 Routing Protocol. In *2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. 1–6. doi:10.23919/SoftCOM52868.2021.9559121

[9] Ana Minaburo, Laurent Toutain, Carles Gomez, Dominique Barthel, and Juan-Carlos Zúñiga. 2020. SCHC: Generic Framework for Static Context Header Compression and Fragmentation. RFC 8724. doi:10.17487/RFC8724

[10] Nsnam. 2025. ns-3 Network Simulator. https://www.nsnam.org/.

[11] C.E. Perkins and E.M. Royer. 1999. Ad-hoc on-demand distance vector routing. In *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*. 90–100. doi:10.1109/MCSA.1999.749281

[12] Charles E. Perkins, John Dowdell, Lotte Steenbrink, and Victoria Pritchard. 2024. *Ad Hoc On-demand Distance Vector Version 2 (AODVv2) Routing*. Internet-Draft draft-perkins-manet-aodvv2-05. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-perkins-manet-aodvv2/05/ Work in Progress.

[13] Francesco Todino, Tommaso Pecorella, and Flavio Esposito. 2025. A Multi-Metric Approach in AODVv2: Enhancing Energy Efficiency and Security for MANETs. In *2025 IEEE 10th International Conference on Network Softwarization (NetSoft)*.

[14] Enrica Zola, Israel Martin-Escalona, Francisco Barceló-Arroyo, and Sergio Machado. 2021. Implementation and analysis of the AODVv2 Routing Protocol in ARM devices. In *2021 International Symposium on Networks, Computers and Communications (ISNCC)*. 1–6. doi:10.1109/ISNCC52172.2021.9615672