A look at routing protocols' PLR for FANETs using ns2

Filipe Freire
FCT, University of Algarve, Portugal
a85493@ualg.pt

ABSTRACT

This study examines the performance of different routing protocols in Flying Ad-Hoc Networks (FANETs), focusing on Packet Loss Ratio (PLR) among unmanned aerial vehicles (UAVs). We compared AODV, OLSR, DSDV, and DSR protocols under simulated conditions while trying to mimic UAV patrol behavior. Results showed that the number of network nodes influences PLR, with reactive protocols, especially DSR, seemingly outperforming proactive ones in lower packet loss ratio. OLSR showed the highest PLR, possibly suggesting an inefficiency in our simulated environment. There are also some highlights to the impact of network dynamics on protocol performance, with AODV showing variability sensitive to these dynamics. Our findings suggest that reactive protocols may offer more resilience in FANET configurations, but we cannot discount limitations related to simulation assumptions and the need for broader and deeper protocol and network condition explorations. Future research should expand on these findings to help peer into routing protocol behaviors for FANETs.

KEYWORDS

Routing Protocols, Wireless Networks, ns2, FANET, AODV, DSDV, OLSR, DSR, UAV

1 INTRODUCTION

In this work we will attempt to look at the simulated behavior of different routing protocols in the context of a Flying Ad-Hoc Network (FANET). FANETs are Ad-hoc networks between unmanned aerial vehicles (UAVs) usually deployed for military and/or civillian purposes, and this realm the dynamics of UAVs introduce complex variables into the communication protocols that govern their interaction [1].

The following routing protocols will be reviewed: AODV [2], OLSR [3], DSDV [4] and DSR [5]. In this report we will only be comparing these through the lens of Packet Loss Ratio (PLR).

2 THE PROBLEM

This study aims to simulate a specific FANET scenario wherein a number of UAVs are dispersed across a large field, all operating at the same altitude.

The primary objective is for the first node to transmit packets at a constant bit rate to the last node, which is farthest from it. See Figure 1 for context.

Given the airborne nature of these nodes, they are not stationary but rather move randomly, simulating a patrol behavior across the field. This scenario might help us simulate an meaningful enough environment to assess the efficiency and reliability, at least in terms of PLR, of different routing protocols under conditions that mimic real-world operational constraints of UAVs in a FANET.

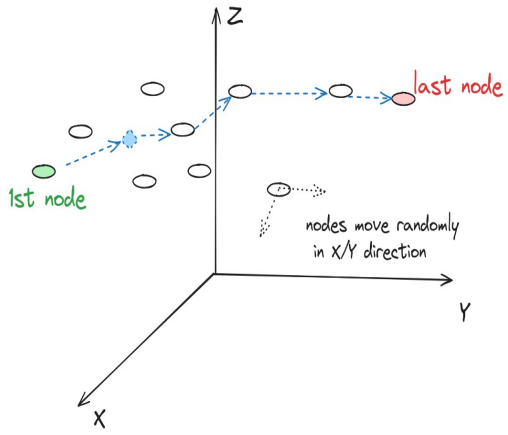

Figure 1: FANET high-level scheme used for simulation

2.1 Splitting the problem

With that in mind we will try to intuit what are the different steps (e.g. "smaller problems") that we can break appart the work into, so it's easier to reproduce. Here's a few steps we can infer of things that we will need to do:

- Find a way that we can setup ns-2 so that we can simulate a 3d scenario more accurately. This will be done by editing some settings and using a patch to ns-2 that adds some 3d capabilities [6].
- Find an accessible way pre-create multiple "flying" mobile nodes, set their position and also their speed and movement direction. This will be described in a later section.
- Find a way in which we can, from any trace file we generate, parse and extract the packet loss ratio (PLR) to compare the routing protocols performance.
- Find a way to use routing protocols that are not fully available by default in ns-2, like for example OLSR [3]. To add this routing protocol into ns-2 we can patch it in using a community implementation [7]

2.2 Similar work

Simulating the behavior of some routing protocols in the context of FANETs has been previously looked at [8], [9], [10].

Most of these and others not go to great lengths sharing any code snippets, reproducible instructions, or breaking down the whole performance evaluation work into smaller problems that would help those not familiar with network simulation in ns-2 reproduce and recreate the results presented in the article.

These articles among others, compare some of the routing protocols that we will compare, but oftentimes do not share a detailed reproduction, only some high-level instructions.

In terms of results shared we might also pose some questions towards their statistical significance as there are no mentions of the number of individual attempts for each simulation configuration, nor is there a detailed explanation of the behavior of the topology and the mobile nodes in the context of a FANET scenario.

There's also usually no explanation into the setup done on ns2 to allow for meangingful FANET simulation. We will go over details in an follow-up question when we go over thesimulation.

It's important to note that there are also some reported warnings to keep in mind as to whether these routing protocols are even suitable for applications like FANET [11] [12], but we will not look at these in the context of this report, which is done for academic coursework purposes in the context of a master's degree program.

2.3 Theory on the routing protocols

Although not the main focus of this report, we will briefly go over the theory of the routing protocols we will be testing. We will be trying two types of routing protocols: **Proactive**, like OLSR and DSDV, and **Reactive**, like DSR and AODV.

Let's go briefly into each:

2.3.1 AODV. AODV [2] stands for Ad hoc On-demand Distance Vector. It is designed to enables dynamic, self-starting, multi-hop routing between participating mobile nodes wishing to establish and maintain an ad hoc network. AODV discovers routes on an on-demand basis using a route discovery process and uses sequence numbers to ensure the "freshness" of routes and to avoid loops.

In theory it is proposed as a robust and efficient choice for wireless networking where the network topology changes frequently. This would be applicable in the case of a FANET in our scenario where the UAVs are constantly changing their position.

2.3.2 DSDV. DSDV [4], or Destination-Sequenced Distance-Vector routing, is a table-driven routing scheme for ad hoc mobile networks based on the Bellman-Ford [13] algorithm. This routing protocol requires each node in the network to maintain a routing table that lists all available destinations, the number of hops to each destination, and a sequence number for the destination set by the destination node. The protocol is suitable for environments with relatively stable networks.

From this last point we can intuit that in an unstable scenario like a FANET where the UAVs are changing their position constantly this routing protocol will not have as good a performance as it would in a more stable scenario. Previous research scenarios [8] [9] [10] appear to go in line with this intuition where AODV performed better than DSDV.

2.3.3 DSR. The Dynamic Source Routing protocol (DSR) [5] is a self-organizing routing protocol designed for use in multi-hop wireless ad hoc networks of mobile nodes. It allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration.

The protocol is based on the concept of source routing, whereby all the routing information is maintained at the mobile nodes. Theoretically we can expect this protocol to adapt well to our FANET simulation scenario.

2.3.4 OLSR. Optimized Link State Routing (OLSR) [3] is routing protocol designed for mobile ad-hoc networks (MANETs). OLSR is designed as a robust solution for dynamic and rapidly changing network environments, such as those encountered in disaster recovery, military communications, and mobile ad-hoc networking scenarios.

One note that will be relevant for us from the lens of simulating this protocol is that upon reading through the specification [3] and after reading through some research (e.g. [14]), we are left with the impression that given its underlying continuous process of maintaining up-to-date topology information, this will result in increased computational overhead and simulation time, especially in larger networks or in scenarios with high mobility like the one we will attempt to simulate.

3 SIMULATION

3.1 Setup

All the simulations ran on ns2 network simulator [15], installed [16] on Ubuntu 22.04 virtual machines in VirtualBox. The usual configuration for the virtual machines was of 8 to 12 virtual CPU cores and roughly 8 GB of RAM.

There are also some extra setup steps required in order to be able simulate FANET-like scenarios in ns2, described in the next section.

3.2 Preparing ns2 for FANET

There two key problems mentioned in a previous section that we will need to solve. The first is adding proper 3D support, since if we are to simulate a FANET we will need to account for 3 dimensions. The second problem is that OLSR routing protocol is not available by default in ns2 so we will need to patch it in.

3.2.1 Adding 3D support to NS2. To add 3D support we will use M2ANET [6], following what was mentioned in [17]:

```
# Backup previous ns-allinone-2.35
cp -r ~/ns-allinone-2.35 ~/ns-allinone-2.35-
    backup

# Download and setup the M2ANET patch:
wget -nc https://www.math.unipd.it/~cpalazzi
    /resources/M2ANET.zip
unzip M2ANET.zip
cd ~/ns-allinone-2.35/ns-2.35/ && patch -p 1
     < ~/M2ANET/ns2_3D.patch
cd ~/ns-allinone-2.35 && ./install
```

To see if it worked, running the script wireless3D.tcl that comes with M2ANET folder should not return any errors:

```
ns ~/M2ANET/wireless3D.tcl
```

There are a few things to keep in mind for FANET simulations in the context of NS2 after installing M2ANET:

- Be sure to set the appropriate queue type for DSR routing protocol `CMUPriQueue`, example `set val(ifq) CMUPriQueue`;
- Be sure to set the propagation as `Propagation/FreeSpace`;
- Be sure to use `load_cube` to account for 3 dimensions in the topography instead of `load_flatgrid`, which requires the M2ANET patch;
- We will need to use `setdest3d` instead of `setdest` method, which also requires M2ANET patch installed;
- Be sure to set Z coordinate / altitude to the mobile nodes;

3.2.2 Adding OLSR support to NS2. To add OLSR routing protocol support we will be applying *F. J. Ros*'s UM-OLSR patch [7] to ns2:

```
#backup previous step
cp -r ~/ns-allinone-2.35 ~/ns-allinone-2.35-
    backup-3d

wget -nc https://downloads.sourceforge.net/
    project/um-olsr/um-olsr/1.0/um-olsr-1.0.
    tgz
tar zxvf um-olsr-1.0.tgz -C ~/

mkdir ~/ns-allinone-2.35/ns-2.35/olsr
cp -r ~/um-olsr/* ~/ns-allinone-2.35/ns-2.35/
    olsr/.

cd ~/ns-allinone-2.35/ns-2.35/ && patch -p 1
    < ~/um-olsr/um-olsr_ns-2.35_v1.0.patch
cd ~/ns-allinone-2.35 && ./install
```

Besides setting routing protocol we will need to configure some `Agent/OLSR` configuration parameters for the simulation to work, for example:

```
# Initialize ns-2 Global Variables
set ns_  [new Simulator]
Agent/OLSR set use_mac_ true
# ...
```

More notes on these can be found over in olsr-ns2 GitHub repository [18] which is an adaptation of UM-OLSR original codebase [19].

3.3 Simulation platform

talk about the general idea of attempting to send packets of one node to another node, while the nodes are moving in random directions.

Then we increase the number of nodes to see how the different routing protocols would handle, taking into account that the nodes are moving around and depending on the routing protocol this might mean there is the risk of loops or the routing tables that guide the hops of a packet between nodes to reach the destination node can vary.

The simulation parameters were chosen as outlined in Table 1.

Table 1: Simulation Parameters and Values

Parameter	Values
Simulator and version	Ns-all-in-one(version 2.35)
NS2 Patches applied	M2ANET for 3d support, UM-OLSR for OLSR support
Channel Type	Wireless
Radio Propagation model	Propagation/FreeSpace
Grid setup	load_cube topography with 5000 units in X, Y and Z
Antenna	Antenna/OmniAntenna
Routing Protocols used	AODV, DSDV, DSR, OLSR
Queue type	Queue/DropTail/PriQueue for AODV, DSDV and OLSR; CMUPriQueue for DSR
Number of nodes	20, 30, 40, 50, 60, 70, 80, 90
Type of Traffic	Application/Traffic/CBR (with first node as Agent/UDP, last node as Agent/Null)
CBR parameters	512 packet size, 0.1 time interval
Mac Layer Protocol	802.11
Simulation duration	600 seconds
Mobile node speed	2 units/sec
Initial distance between nodes	200 units in X and Y axis
Altitude	400 units in Z axis
Max distance for node's new random position	5 units
Time interval between new random position for nodes	15 seconds
Random seed setup	Ran each routing protocol for each number of nodes, individually with 30 random seeds (1 to 30)

A few parameters that are not mentioned in the table but are key for the simulation are how the mobile nodes are positioned in the topography and how they move through time based on a random seed at each individual execution.

For positioning the nodes at the start of the simulation we split the nodes into a grid like pattern, distanced between each other on X and Y axis by an initial distance and set with the initial altitude value mentioned in the Table 1.

```
# Calculate the number of rows and columns
    for the nodes
set rows [expr {int(sqrt($num_nodes))}]
```

```
set cols [expr {int(ceil(double($num_nodes) /
     $rows))}]

# Calculate division size based on the
    constraints
set division_size_x $max_distance
set division_size_y $max_distance

# Calculate starting positions to center the
    grid
set start_x [expr {($gridSize - ($cols - 1) *
     $division_size_x) / 2.0}]
set start_y [expr {($gridSize - ($rows - 1) *
     $division_size_y) / 2.0}]

# Create nodes and position them in the grid
for {set i 0} {$i < $num_nodes} {incr i} {
    set node_($i) [$ns_ node]
        $node_($i) random-motion
            $random_motion
        $ns_ initial_node_pos $node_($i)
            $node_size

    # Calculate grid positions of i-th node
    set col [expr {$i % $cols}]
    set row [expr {$i / $cols}]

    set x_pos [expr {$start_x + $col *
        $division_size_x}]
    set y_pos [expr {$start_y + $row *
        $division_size_y}]

    # Set initial position at time 0
    $node_($i) set X_ $x_pos
    $node_($i) set Y_ $y_pos
    $node_($i) set Z_ $altitude
    $ns_ at 0.0 "$node_($i) setdest3d $x_pos
        $y_pos $altitude 0"
```

Throughout the simulation the nodes are not stationary. They move randomly in the X and Y axis to new positions, calculated using a helper procedure:

```
# Calculate a new position within the move
    range, ensuring it's within the grid
proc calculateNewRandomPosition {x_pos y_pos
    max_move gridSize buffer} {
    set move_x [expr {rand() * (2 * $max_move
        ) - $max_move}]
    set move_y [expr {rand() * (2 * $max_move
        ) - $max_move}]
    set new_x [expr {max($buffer, min($x_pos
        + $move_x, $gridSize - $buffer))}]
    set new_y [expr {max($buffer, min($y_pos
        + $move_y, $gridSize - $buffer))}]
    return [list $new_x $new_y]
}
```

These new calculated random positions are set at different time intervals, and following a max distance also configured with an initial value mentioned in the Table 1.

```
# every time_step seconds set another random
    destination
for {set t 0} {$t < $time_duration} {incr t
    $time_step} {
    # Calculate a new position within the
        grid for the node to move to
    set new_position [
        calculateNewRandomPosition $x_pos
        $y_pos $max_move $gridSize $buffer]
    set new_x [lindex $new_position 0]
    set new_y [lindex $new_position 1]
    # Schedule the node to move to the new
        position at next time step
    $ns_ at $t "$node_($i) setdest3d $new_x
        $new_y $altitude $speed"
}
```

There is also a buffer variable defined, hardcoded at the time of writing to 25 units, that is used when calculating new random positions in a way such as to prevent the nodes of being directed out of the topography grid limits.

In terms of network traffic, the first node will attempt to send CBR packets to the last node, the packets defined a size and interval mentioned in the Table 1.

```
set udp [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp
set null [new Agent/Null]
$ns_ attach-agent $node_([expr $val(nn) - 1])
     $null
$ns_ connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ $cbr_packet_size
$cbr set interval_ $cbr_time_interval
$cbr attach-agent $udp
$ns_ at 2.0 "$cbr start"
$ns_ at $val(stop) "$cbr stop"
$ns_ at $val(stop) "stop"
$ns_ at $val(stop).01 "puts \"NS EXITING...\"
    ; $ns_ halt"
```

For parsing the trace files to extract the packet loss ratio, we used the following GNU awk [20] script:

```
BEGIN {
        receivedFanet = 0
        sentFanet = 0
        ratioFanet = 0
```

```
        packetLoss = 0
}
{
        if($1 == "s" && $3 == "_0_" && $4 ==
            "AGT") {
                sentFanet++
        }
        if($1 == "r" && $3 == receiverNode &&
             $4 == "AGT") {
                receivedFanet++
        }
}
END {
        ratioFanet = ((sentFanet -
            receivedFanet) / sentFanet) * 100
        packetLoss = sentFanet -
            receivedFanet
        printf("\nrp␣%s,␣nn␣%d,␣random␣seed␣%
            d,␣Received:␣%d␣|␣Sent:␣%d␣|␣
            Dropped:␣%d,␣PLR:␣%f%", rp, nn,
            seed, receivedFanet, sentFanet,
            packetLoss, ratioFanet)
}
```

3.4 Results

In our exploration of network performance through simulation, several patterns have emerged that illuminate the behavior of different routing protocols under varying conditions. Firstly, it has become evident that the number of nodes within the network directly influences the packet loss ratio. Specifically, an increase in nodes tends to correlate with a heightened probability of packet loss (*see Figure 2 for a graphical representation of the average PLR*). This observation underscores the complexities involved in scaling network infrastructures, particularly in environments where maintaining low packet loss would be critical, like in some critical UAV scenarios of civilian or military nature.

When comparing the performance of reactive and proactive routing protocols, our simulations reveal a distinct advantage in favor of reactive protocols. Among the protocols evaluated, the Optimized Link State Routing (OLSR) protocol exhibited the highest packet loss ratio, making it the least efficient in our scenarios (*see Figure 7 for a graphical representation of the average PLR*).

Following OLSR, the Destination-Sequenced Distance-Vector (DSDV) protocol also demonstrated suboptimal performance, securing its position as the second least favorable option in terms of packet loss mitigation (*see Figure 4 for a graphical representation of the average PLR*).

These findings suggest that, within the confines of our simulation parameters, reactive protocols, like DSR and AODV, may offer more resilient network configurations than their proactive counterparts.

DSR proved to be, in the context of our simulation, the best routing protocol in terms of having the lowest packet loss ratio, as can be seen in Figure 6.

An intriguing variability was observed with the Ad hoc On-demand Distance Vector (AODV) routing protocol, which displayed

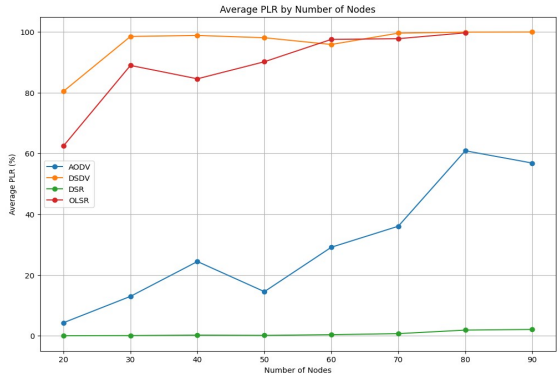

Figure 2: Average PLR of routing protocols over number of nodes

a notable increase in standard deviation of packet loss ratios as the number of nodes escalated (see figure 3). We suspect this indicates that AODV's performance is highly sensitive to the underlying network dynamics, such as the random seed used to simulate node movement.

Consistent with preliminary expectations, the overall behavior of the evaluated protocols aligned with our theoretical understandings of their mechanisms described in a previous section.

This provides a measure of validation for our simulation approach and the assumptions underpinning it. However, it is crucial to acknowledge the limitations of our study. The realism of our results may be constrained by several factors, including:

- the use of default parameters for the OLSR protocol
- the lack of specific configuration for signal and transmission properties
- and the absence of more trials with varied distances between nodes.

Although exploring scenarios with static nodes could potentially offer insights into optimal configurations for minimizing packet loss, our aim was to balance the simulation conditions with those of realistic scenarios likely to be encountered in Flying Ad-hoc Networks (FANETs).

The readers can refer to also so the following figures for a portrayal of quartiles for each routing protocol:

- DSDV - see Figure 4
- DSR - see Figure 6
- AODV - see Figure 5
- OLSR - see Figure 7

3.4.1 Side-note on OLSR. When running the OLSR simulation it was noticeable that it ran significantly slower for any given individual execution compared to another routing protocol's counterpart. This is somewhat expected due to what was mentioned in a previous section, where OLSR is more intensive than other routing protocols. Readers of this report will notice that we are missing values for 90 nodes for OLSR. These were not accomplished due to time constraints for delivering the report, as it would take 1-2 hours to simulate per iteration (with a single random seed).

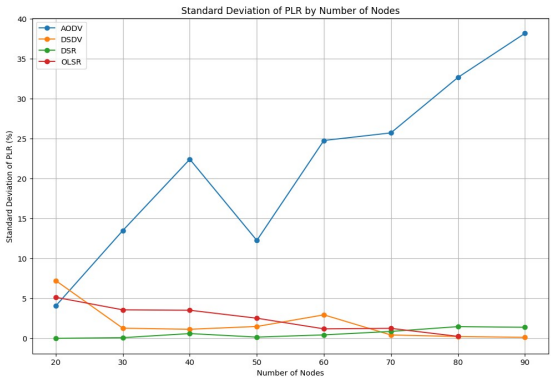

Figure 3: Standard deviation of PLR of routing protocols over number of nodes

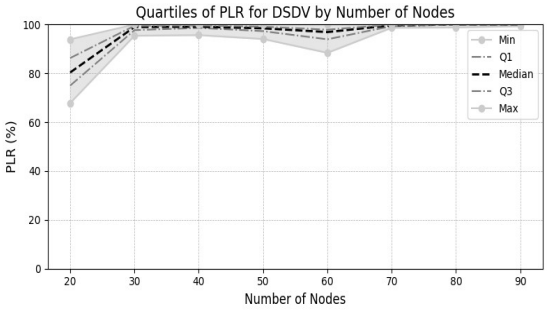

Figure 4: Quartiles for DSDV over number of nodes

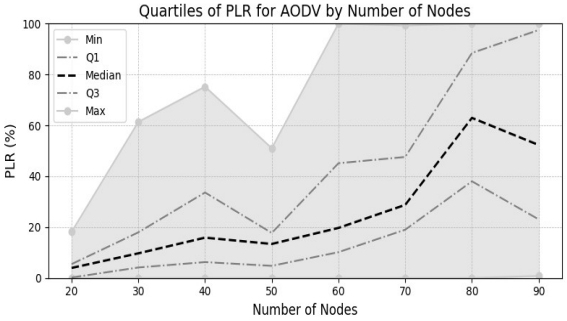

Figure 5: Quartiles for AODV over number of nodes

This time constraint can be minimized by running the ns2 simulations in parallel across the number of CPU cores available. One example of how this was achieved in our case was with resource to xargs [21] and nproc [22], like show bellow, as an example, in a custom makefile target:

```
fanet:
 @seq $(MIN_NODES) $(NODES_INCREMENT) $(
     MAX_NODES) | while read num; do \
```

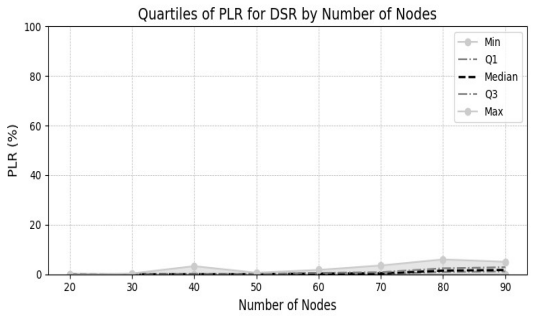

Figure 6: Quartiles for DSR over number of nodes

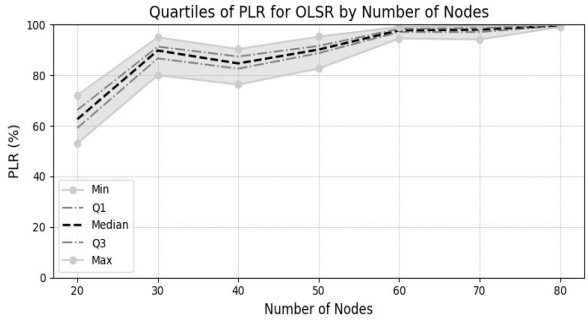

Figure 7: Quartiles for OLSR over number of nodes

```
    seq 1 $(MAX_RANDOM_SEEDS) | while read seed
        ; do \
     echo AODV $$num $(TIME_DURATION) $(
         GRID_SIZE) $$seed $(DIST_BETWEEN_NODES
         ) $(RESULTS_FOLDER); \
     echo DSDV $$num $(TIME_DURATION) $(
         GRID_SIZE) $$seed $(DIST_BETWEEN_NODES
         ) $(RESULTS_FOLDER); \
     echo DSR $$num $(TIME_DURATION) $(
         GRID_SIZE) $$seed $(DIST_BETWEEN_NODES
         ) $(RESULTS_FOLDER); \
    done; \
done | xargs -P `nproc` -I {} bash -c 'ns
    fanet-lab2-filfreire.tcl {} > /dev/null'
```

4 CONCLUSION

While this work may contribute some insights into the performance of different routing protocols within a simulated FANET environment configuration, the scope of our study is delimited by certain assumptions and configurations.

These assumptions and configurations might have negatively impacted the results for the evaluated protocols. Reactive protocols, particularly DSR, apparently demonstrated superior performance in minimizing packet loss, while the Optimized Link State Routing (OLSR) protocol exhibited the highest PLR, marking it as the least efficient.

This point is somewhat suspicious, as it is not according to what the protocol is seemingly designed for [3], we were expecting it to be more robust. We can narrow this down for now to perhaps wrongful simulation platform configuration for the particular case of OLSR.

Additionally, the Ad hoc On-demand Distance Vector (AODV) protocol showed significant variability in performance, suggesting either an actual high sensitivity to network dynamics, or also wrongful simulation configuration like for OLSR. But we are not sure at the time of writing.

These findings underscore the complexities of simulating FANETs and follow-up research efforts would need to consider a more exhaustive exploration of protocol parameters and network conditions to fully understand the implications of our observations.

REFERENCES

[1] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (fanets): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[2] S. R. Das, C. E. Perkins, and E. M. Belding-Royer, "Ad hoc On-Demand Distance Vector (AODV) Routing." RFC 3561, July 2003.

[3] T. H. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)." RFC 3626, Oct. 2003.

[4] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," *ACM SIGCOMM computer communication review*, vol. 24, no. 4, pp. 234–244, 1994.

[5] D. B. Johnson, D. A. Maltz, J. Broch, *et al.*, "Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks," *Ad hoc networking*, vol. 5, no. 1, pp. 139–172, 2001.

[6] N. Mahmood, J. DeDourek, and P. Pochec, "M2anet simulation in 3d in ns2," *Proc. SIMUL*, pp. 24–28, 2014.

[7] F. Ros, "Um-olsr, an implementation of the olsr (optimized link state routing) protocol for the ns-2 network simulator," *Software package retrieved from https://sourceforge.net/projects/um-olsr/*, 2007.

[8] K. Singh and A. K. Verma, "Experimental analysis of aodv, dsdv and olsr routing protocol for flying adhoc networks (fanets)," in *2015 IEEE international conference on electrical, computer and communication technologies (ICECCT)*, pp. 1–4, IEEE, 2015.

[9] A. Nayyar, "Flying adhoc network (fanets): simulation based performance comparison of routing protocols: Aodv, dsdv, dsr, olsr, aomdv and hwmp," in *2018 international conference on advances in big data, computing and data communication systems (icABCD)*, pp. 1–9, IEEE, 2018.

[10] N. Megha and B. Mallikarjun, "Performance evaluation of routing protocol for fanet using ns2," *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 7, pp. 1128–1132, 2020.

[11] A. Bujari, O. Gaggi, C. E. Palazzi, and D. Ronzani, "Would current ad-hoc routing protocols be adequate for the internet of vehicles? a comparative study," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3683–3691, 2018.

[12] A. Bujari, C. E. Palazzi, and D. Ronzani, "A comparison of stateless position-based packet routing algorithms for fanets," *IEEE Transactions on Mobile Computing*, vol. 17, no. 11, pp. 2468–2482, 2018.

[13] R. Bellman, "On a routing problem," *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[14] O. K. Sahingoz, "Networking models in flying ad-hoc networks (fanets): Concepts and challenges," *Journal of Intelligent & Robotic Systems*, vol. 74, pp. 513–527, 2014.

[15] K. Fall, K. Varadhan, *et al.*, "The ns manual (formerly ns notes and documentation)," *The VINT project*, vol. 47, pp. 19–231, 2005.

[16] F. Freire, "personal guide for installing ns2 — filfreire.com." https://filfreire.com/ns2/. [Accessed 18-03-2024].

[17] "Position-based Routing Protocols - NS-2 - Drones - Flying Ad-Hoc Network - FANET - DANET — math.unipd.it." https://www.math.unipd.it/~cpalazzi/FANET-routing.html. [Accessed 17-03-2024].

[18] "GitHub - wevertoncordeiro/olsr-ns2: OLSR (RFC 3626 + ETX / ML / MD variants) for NS-2 — github.com." https://github.com/wevertoncordeiro/olsr-ns2. [Accessed 17-03-2024].

[19] "UM-OLSR — sourceforge.net." https://sourceforge.net/projects/um-olsr/. [Accessed 17-03-2024].

[20] "The GNU Awk User&x2019;s Guide — gnu.org." https://www.gnu.org/software/gawk/manual/gawk.html. [Accessed 18-03-2024].

[21] "xargs(1) - Linux manual page — man7.org." https://www.man7.org/linux/man-pages/man1/xargs.1.html. [Accessed 18-03-2024].

[22] "nproc(1) - Linux manual page — man7.org." https://man7.org/linux/man-pages/man1/nproc.1.html. [Accessed 18-03-2024].