Journal Homepage: www.ijrpr.com ISSN: 3049-0103 (Online)

International Journal of Advance Research Publication and Reviews

Vol 03, Issue 10, pp 890-897, October, 2026

Two Different Timers Govern Neighbor Detection in Link-State Ad Hoc Routing

Dina Hisham Ahmad [1,*]

[1]Independent Researcher (B.Sc., University of Mosul; M.Sc., Northern Technical University), Mosul, Iraq
[*]uomcoed@gmail.com

ABSTRACT :

OLSRv1 ties the validity time in each HELLO message to the HELLO interval; both the OLSR specification and the ns-3 model set it to three times that interval. We show that this single setting hides two separate timers: one governs how fast a node notices a neighbor leaving, the other how fast it notices one returning. In ns-3.49 timestamp traces, a departed neighbor is purged when the validity time of its last HELLO runs out, with observed and predicted purge times differing by 0.03 s on average across 1,440 observations. Departure detection takes about the hold time minus half a HELLO interval, whereas a returning neighbor is found after about 1.5 intervals whatever the hold time. This explains why a gap-triggered adaptive HELLO controller gave no gain over stock OLSR in 600 runs. We separated the two parameters (HELLO every 3 s, hold time 6 s) and tested them in a prespecified study of 900 runs. Against stock OLSR, control bytes fell by about 10% under churn and node outage and by 7% in stable networks, with no measurable loss in delivery ratio. On fading channels the setting cost about 8 points of delivery ratio and should be avoided.

Keywords: OLSR, HELLO interval, Neighbor hold time, MANET, ns-3, Neighbor discovery

Introduction

Mobile ad hoc networks (MANETs) have no fixed infrastructure, so each node has to find its neighbors and keep its routes current by itself (Corson & Macker, 1999). Reactive protocols such as AODV (Perkins et al., 2003) and DSR (Johnson et al., 2007) build a route only when one is needed, and several have been compared under mobility (Broch et al., 1998). Proactive protocols such as DSDV (Perkins & Bhagwat, 1994) and OLSR (Clausen & Jacquet, 2003; Jacquet et al., 2001) keep routes ready at all times and pay for it with periodic control traffic. For these protocols neighbor discovery is the foundation, since every later routing decision rests on the neighbor table.

OLSR version 1 (OLSRv1) tracks its neighbors through periodic HELLO messages. Each HELLO carries a validity time that tells receivers how long to keep the sender in their tables. RFC 3626 recommends three HELLO refresh intervals (Clausen & Jacquet, 2003), and the ns-3 OLSR model (Riley & Henderson, 2010) derives the advertised value from its HelloInterval attribute in the same way. A single parameter thus controls two unrelated things: how many control bytes a node sends, and how long stale neighbor state survives. The later neighborhood discovery protocol (NHDP) keeps the two as separate parameters (Clausen et al., 2011), so the coupling in OLSRv1 looks like a convention rather than a requirement. Nothing in the message format forces it.

A natural idea is to adapt the HELLO rate: send faster when the neighborhood seems to be changing, slower when it is calm. Many schemes do this (Section 2). We tried it first, with a controller that reacts to a gap in the expected HELLO arrivals, in a frozen confirmatory experiment of 600 runs. It did not improve the delivery ratio over stock OLSR, and at first we did not understand why. The explanation turned out to be short. The timer that removes a departed neighbor is not the one the controller was speeding up.

This paper explains that negative result and then tests what the explanation suggests. Timestamp traces from ns-3.49 show that departure detection follows the hold time advertised in the last HELLO, while detection of a returning neighbor follows the HELLO interval. That gives a two-timer model of neighbor detection in OLSRv1, and it also shows why speeding up the sender cannot shorten departure detection. We then decouple the two parameters and measure the effect in a prespecified study of 900 runs. Decoupling helps in stable, churn, and outage scenarios and hurts on fading channels. The simulation code, a one-attribute ns-3 patch, the protocol written before the runs, and every raw result, including the null and negative ones, are released with the paper.

The rest of the paper is organized as follows. Section 2 reviews related work, Section 3 presents the model, Section 4 describes the method, Section 5 gives the results and discussion, and Section 6 concludes.

Related work

2.1 Neighbor sensing in OLSR and its standards

In OLSRv1 a node learns its symmetric neighbors from HELLO messages. A link tuple expires when the validity time carried by the last HELLO from that neighbor runs out. Because the validity time is a field of the message, the receiver simply follows whatever the sender advertises. RFC 3626 recommends three times the HELLO refresh interval (Clausen & Jacquet, 2003), but nothing in the message format ties the two together. NHDP (Clausen et al., 2011), on which OLSRv2 is built (Clausen et al., 2014), keeps the HELLO interval and the hold times as separate parameters. In the ns-3.49 model the advertised neighbor validity is three times the current HelloInterval attribute, and HELLO emission is jittered (Riley & Henderson, 2010).

Early OLSR experiments showed that jitter on periodic control messages reduces collision losses, and that naive neighbor sensing can accept a bad link at the edge of radio range (Clausen et al., 2001). Link-quality metrics such as ETX (De Couto et al., 2003) address a related problem. The point matters for us too, because a short hold time makes a lossy link more likely to be dropped.

2.2 Studies of HELLO and validity-time settings

Several studies vary the OLSR timers directly. Huang et al. (2006) tuned the refresh intervals in ns-2 and found that the HELLO interval matters more than the other message intervals, with the effect growing with mobility and node density. They note that the default hold time is three times the HELLO interval, but they do not separate the two. Hiyama et al. (2013) ran a real testbed and varied both the HELLO send interval and the validity time, measuring throughput. Razafindralambo and Mitton (2007) analyzed, in theory and in simulation, how the HELLO parameters interact with a realistic MAC layer. Ikeda et al. (2009) compared testbed and ns-2 results for OLSR and saw packet loss in the experiments that the simulation did not show; they attribute it to environmental effects and traffic interference.

The closest earlier work to our decoupling is the thesis of Jovel (2014), who studied the validity time relative to the HELLO interval. He reports that packet loss rises when the validity time falls below twice the interval, and that a validity of twice the interval gave little loss in his settings. He ascribes the loss to neighbor flapping, where an in-range neighbor is repeatedly added to and evicted from the neighbor sets of other nodes. We use the same mechanism to explain our fading results: a hold factor of two lowered the delivery ratio by 8 to 9 points there, although it cost nothing in stable networks. So the safe hold factor seems to depend on how lossy the channel is. Our work differs in what it measures and tests. We record purge and discovery times at timestamp level, so departure and arrival detection can be assigned to different timers; we evaluate decoupled settings in a prespecified study with several scenarios; and we test directly why sender-side adaptation fails.

2.3 Adaptive HELLO schemes

A larger body of work adapts the HELLO emission rate. The turnover-based protocol TAP (Ingelrest et al., 2007) lets each node adjust its HELLO frequency according to how many new neighbors it sees, without needing speed information. ARH (Li et al., 2011) predicts node positions with an autoregressive model and sends a HELLO only when the prediction error is large; its authors report a lower HELLO rate than TAP at similar neighborhood accuracy. Ernst and Martini (2012) adapt the interval in NHDP and coordinate it between nodes, aiming for fast detection of link changes at lower overhead. He et al. (2013) focus on energy efficiency, and Mitton et al. (2024) apply the turnover idea to vehicular networks. For OLSR specifically, Xu et al. (2024) adjust the HELLO interval by watching changes in the local link information base and test the result on UAVs.

These schemes mainly change when a node sends HELLO messages. As far as we can tell from their descriptions, they leave the validity time carried by each message untouched, and that value fixes how long a receiver keeps a neighbor after the last HELLO. Our traces show that in OLSRv1 this is exactly what decides departure detection. For schemes aimed at faster detection, the benefit should therefore be limited to shortening the age of the last HELLO before the event. For schemes aimed at lower overhead, the advertised validity time determines how much staleness the saving costs. This agrees with the negative result of our own gap-triggered controller (Section 5.2).

2.4 Simulation practice and positioning

Simulation studies of MANET protocols have a mixed record. Kurkowski et al. (2005) surveyed the MobiHoc proceedings from 2000 to 2005 and found significant shortfalls, and Andel and Yasinsac (2006) argue that a lack of rigor threatens the credibility of published MANET simulation results. We tried to follow the advice these critiques imply: a separate process and fixed random streams for every run, identical traffic and topology schedules across settings, a protocol written before the runs, and reporting of every result.

Among the works we reviewed, none separates departure from arrival detection with timestamp-level evidence, and none evaluates a decoupled hold time inside an OLSRv1 implementation in prespecified scenarios that include fading. Our review is not exhaustive. Our aim is also narrower than HELLO tuning in general: we separate the two detection events in a controlled way and measure what decoupling the timers costs and gains.

The two-timer model

Let h be the HELLO interval, V the advertised validity time (the hold time), and $k = V / h$ the hold factor. Stock OLSR uses $k = 3$.

Departure. Suppose a neighbor leaves at time t_{dep}. The last HELLO a receiver heard from it was sent at $t_{last} = t_{dep} - a$, where the age a lies between 0 and about h. Its mean is about $h/2$ when the departure is not aligned with the HELLO timer. The receiver purges the neighbor at $t_{last} + V$, whatever the departed node does afterwards. The detection delay D_{dep} is therefore given by (1), and its mean by (2).

$$D_{\text{dep}} = V - a \tag{1}$$

$$\mathrm{E}[D_{\text{dep}}] \approx V - \frac{h}{2} \tag{2}$$

Arrival. A node returns at a random time. Its first HELLO after the return reaches the receiver after about $h/2$ on average. The receiver lists the node as symmetric only after one more HELLO from it that includes the receiver in its heard list, which takes about one more interval. The mean arrival delay D_{arr} is therefore

$$\mathrm{E}[D_{\text{arr}}] \approx 1.5h \tag{3}$$

and it does not depend on V.

Three consequences follow. First, once a neighbor has left, nothing about its sending rate can move the purge time; sender-side acceleration can only shorten the age a of the last HELLO before the event. Second, with the stock coupling $V = 3h$, lowering h to find arrivals sooner also lowers V, and raising h to save bytes also raises V. Designers have one dial for two jobs. Third, a hold factor below about 3 shortens departure detection but makes false purges more likely, because fewer consecutive HELLO losses are needed to expire a live neighbor.

False purges. Shortening the hold time has a price. Suppose HELLOs go out about h apart and the last one received arrived at time 0. The neighbor expires at kh, so every HELLO sent at h, $2h$, and so on up to just before kh has to be lost to cause a false purge. That is about $\lceil k \rceil - 1$ losses in a row: one for $k = 2$, two for $k = 3$. With independent losses of probability p, the chance is roughly $p^{\lceil k \rceil - 1}$ per expiry window. This is only a rule of thumb. Fading losses come in bursts, and in Section 5.3 the rule turned out too optimistic for $k = 2.5$, so on lossy channels we treat $k \geq 3$ as the safe choice.

Method

4.1 Simulation setup

All experiments use ns-3.49 with its OLSRv1 model and IEEE 802.11b at 1 Mbit/s in ad hoc mode, on grid topologies with 35 m spacing and a maximum range of 65 m. Path loss is log-distance with an exponent of 3.0, or 3.5 together with Nakagami fading in the harsh scenarios; the reference loss is 40.05 dB. The OLSR TC interval is 5 s. Application traffic is up to five UDP flows (256-byte payload every 0.25 s) between fixed node pairs. Measurements cover 25 s to 90 s. Topology changes happen at 45 s and are reverted at 70 s. Every run is a separate process with a fixed random-stream assignment on the Wi-Fi devices, the channel, and OLSR, and the traffic and topology schedules are identical across timer settings.

4.2 Timer settings

We use six static settings, written H(hello)/hold: the stock default H2/6 s, then H3/6 s, H3/7.5 s, H3/9 s, H5/12.5 s, and the stock slow setting H5/15 s. The three settings whose hold factor differs from 3 need a one-attribute patch (NeighbHoldFactor) that makes the advertised validity time a configurable multiple of the HELLO interval. With the default factor of 3 the model behaves exactly like stock ns-3.

4.3 Experiment A: timestamp trace

This experiment tests the mechanism. A 5 by 5 grid carries OLSR control traffic only, with no data traffic and no fading. The central node, which has eight in-range neighbors, leaves the network at a time drawn between 40 s and 70 s and returns 30 to 60 s later. The times come from low-discrepancy sequences of the seed and are identical across settings, so they are not aligned with the HELLO timer grid. Each neighbor logs the last HELLO received before the departure together with its advertised validity time, polls its symmetric neighbor set every 50 ms, and records the purge time, the first HELLO after the return, and the time at which the returning node is listed again. Six settings with 30 seeds each (4001 to 4030) gave 1,440 observer records. The unit of analysis for means and confidence intervals is the seed, because the eight observers of one run share the same leaver and the same timing.

4.4 Experiment B: system-level study

Experiment B was prespecified in a protocol written before any run. It crosses 6 settings, 5 scenarios (stable, fading, churn, outage, and stress), 3 network sizes (25, 50, and 100 nodes), and 10 new seeds (2003, 2011, 2017, 2027, 2029, 2039, 2053, 2063, 2069, and 2081), which gives 900 runs. Churn moves every seventh node by (70 m, 25 m). Outage moves every thirteenth node 2,000 m away, and node 0, a flow source, is isolated for every setting. Stress combines both with fading. The outcomes are the packet delivery ratio (PDR, attempt-based), the OLSR control bytes per node per minute, and an evaluator-only count of stale and missing symmetric-neighbor entries based on node positions. Comparisons are paired against the stock default by seed and size, with a bootstrap over seeds for 95% intervals.

4.5 Experiment C: earlier adaptive controller

Experiment C is the frozen confirmatory study that came first. It was run before Experiment B but is reported after Experiment A because it motivates B. It compares four policies over 5 scenarios, 3 sizes, and 10 seeds (600 runs): the stock default (HELLO 2 s), the stock slow setting (5 s), a pilot that reacts to neighbor-set changes, and a gap-hysteresis controller. The last one switches from a 5 s to a 2 s HELLO interval for 12 s (cooldown 15 s) when a neighbor's HELLO gap exceeds two expected periods plus 0.35 s. The protocol was frozen before the run and not tuned afterwards.

4.6 Reporting rules

All runs of every experiment are reported, including negative and null results. Parameters were never changed after seeing results within an experiment; a change started a new round with new seeds. About 50 paired comparisons were made in Experiment B, and we did not correct for multiplicity, so intervals that exclude zero are treated as exploratory. An early version of the trace used a fixed departure time aligned with the HELLO timer grid and gave arrival delays that were nearly constant across seeds. We removed this phase locking by randomizing the event times, and only the randomized trace is reported.

Results and discussion

5.1 Experiment A: the purge time is fixed by the last HELLO

Across the 1,440 observations the observed purge time matches the predicted expiry, which is the reception time of the last HELLO plus its advertised validity time. The mean difference is 0.025 to 0.031 s per setting and the maximum is 0.050 s, equal to the polling resolution (Fig. 1a). The advertised validity was exactly 6, 7.5, 9, 12.5, and 15 s for the corresponding settings. The mean detection delays follow the model closely (Table 1, Figs. 1b and 1c).

Table 1 - Mean detection delays over 30 seeds (95% bootstrap CI in brackets) against the two-timer model

Setting	Hello (s)	Hold (s)	Purge delay (s)	Model: hold − hello/2	Arrival delay (s)	Model: 1.5 × hello
H2/6 (stock)	2	6	4.98 [4.76, 5.20]	5.0	3.07 [2.91, 3.23]	3.0
H3/6	3	6	4.44 [4.09, 4.78]	4.5	4.68 [4.42, 4.94]	4.5
H3/7.5	3	7.5	6.01 [5.66, 6.36]	6.0	4.66 [4.39, 4.92]	4.5
H3/9	3	9	7.42 [7.07, 7.77]	7.5	4.60 [4.36, 4.84]	4.5
H5/12.5	5	12.5	9.88 [9.33, 10.41]	10.0	7.52 [7.12, 7.91]	7.5
H5/15 (stock slow)	5	15	12.38 [11.82, 12.93]	12.5	7.61 [7.19, 8.01]	7.5

A least-squares fit over the 180 seed-level means gives purge delay = 0.06 + 0.996 × hold − 0.526 × hello and arrival delay = 0.16 − 0.013 × hold + 1.519 × hello. The coefficients sit close to the model values of (1, −0.5) and (0, 1.5). At an equal hold time of 6 s, the 3 s HELLO setting purges a departed neighbor 0.55 s earlier than the 2 s setting (paired difference, 95% CI −0.89 to −0.20), but it takes 1.62 s longer to list a returning neighbor (CI 1.33 to 1.90).

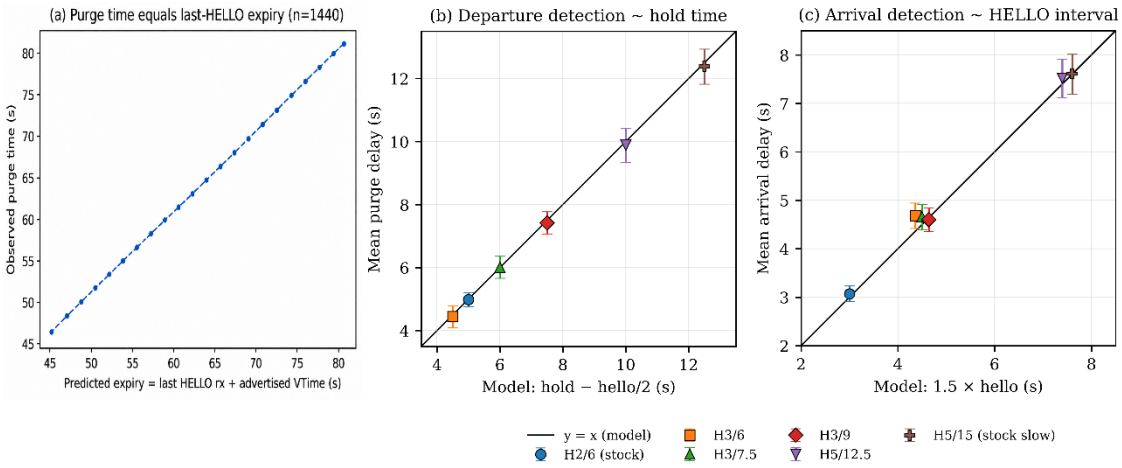


Fig. 1 - (a) Observed against predicted purge time. (b) Purge delay against hold − hello/2. (c) Arrival delay against 1.5 × hello. Error bars are 95% bootstrap intervals over seeds.

5.2 Experiment C: why sender-side adaptation did not help

In the 600-run confirmatory study the gap-hysteresis controller did not improve on stock OLSR (Table 2). Under churn its PDR (76.5%) was close to that of the stock slow setting (75.6%), which it resembles whenever it is not in fast mode, and 10 points below the stock default (86.5%). The controller switched to fast mode 18,172 times in total, yet this changed little. The model says why: after a neighbor has left, receivers purge it using the validity time of its last HELLO, and that HELLO was advertised at the slow setting.

Table 2 - Experiment C, mean over all scenarios, sizes, and seeds

Policy	PDR (%)	Control bytes per node per min	PDR under churn (%)
Stock default (H2)	66.5	9,948	86.5
Stock slow (H5)	63.0	8,756	75.6
Pilot	63.1	9,440	77.6
Gap hysteresis	61.7	8,894	76.5

5.3 Experiment B: decoupling hold time from HELLO rate

All 900 runs were valid. The time that stale symmetric entries survive follows the hold time and not the HELLO interval. Under churn the mean stale entry-seconds were 672 (hold 6 s, H2), 632 (hold 6 s, H3), 892 (7.5 s), 1,094 (9 s), 1,213 (12.5 s), and 1,578 (15 s). The same ordering appears under outage and stress.

Table 3 compares H3/6 s with the stock default. The two have the same hold time, but H3/6 s sends one third fewer HELLO messages. The pooled PDR of the stock default was 98.3% in stable networks, 34.2% under fading, 86.0% under churn, 86.7% under outage, and 23.8% under stress.

Table 3 - H3/6 s minus stock default (H2/6 s), paired by seed and size; 95% CI over seeds in brackets

Scenario	PDR difference (pp)	Control bytes difference (%)
Stable	+0.9 [−1.1, +3.1]	−7.4
Churn	+2.7 [+0.4, +5.9]	−10.5
Outage	+1.0 [−0.4, +2.6]	−10.0
Fading	−8.6 [−11.0, −6.4]	−33.8
Stress	−7.9 [−9.6, −6.3]	−39.7

In the stable, churn, and outage scenarios the decoupled setting saves 7 to 10% of control bytes without a measurable loss of delivery ratio. Under churn the PDR gain depends on network size: −1.4 pp at 25 nodes (higher in 2 of 10 seeds), +4.4 pp at 50 nodes (8 of 10), and +5.1 pp at 100 nodes (9 of 10). The byte saving falls from 17.8% at 25 nodes to 4.7% at 100 nodes. Under outage the PDR differences are −0.1, +1.6, and +1.5 pp at 25, 50, and 100

nodes. On fading channels (fading and stress) the same setting lowers PDR by about 8 points. The large byte reductions there (34 to 40%) come with lost links and do not indicate efficiency: fewer neighbors are listed, and a hold factor of 2 leaves almost no margin against a lost HELLO.

Slower stock settings save more bytes than H3/6 s in stable networks (H5/15 s: −17%), but under churn they lose 11.4 pp of PDR (CI −13.2 to −8.8), so in that regime they sit on a worse trade-off than the decoupled setting. Figure 2 plots the PDR difference against the byte difference for every setting and scenario, and Table 4 lists the numbers, including the intermediate decoupled settings (H3/7.5 s, H3/9 s, and H5/12.5 s). Note the fading rows: H3/9 s, which keeps a hold factor of 3, loses nothing there, while H3/7.5 s and H5/12.5 s, with a factor of 2.5, lose 3 to 4 points.

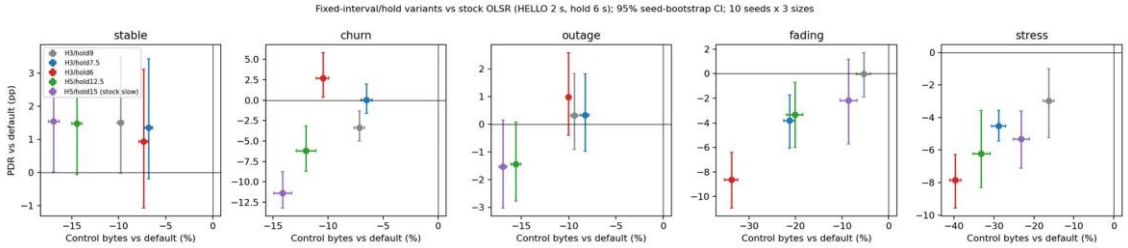


Fig. 2 - PDR difference against control-byte difference relative to the stock default for each setting and scenario, with 95% bootstrap intervals.

Table 4 - Paired results against the stock default for all settings and scenarios, pooled over sizes; 95% bootstrap CI over seeds for PDR

Scenario	Setting	PDR difference (pp) [95% CI]	Bytes difference (%)
Stable	H3/9	+1.50 [−0.03, +3.49]	−9.8
Stable	H3/7.5	+1.35 [−0.20, +3.43]	−6.8
Stable	H3/6	+0.93 [−1.08, +3.11]	−7.4
Stable	H5/12.5	+1.48 [−0.06, +3.58]	−14.5
Stable	H5/15	+1.54 [+0.01, +3.62]	−16.9
Churn	H3/9	−3.35 [−4.98, −1.29]	−7.2
Churn	H3/7.5	+0.04 [−1.62, +1.98]	−6.5
Churn	H3/6	+2.71 [+0.35, +5.86]	−10.5
Churn	H5/12.5	−6.16 [−8.69, −3.15]	−12.0
Churn	H5/15	−11.35 [−13.22, −8.75]	−14.1
Outage	H3/9	+0.32 [−0.91, +1.82]	−9.4
Outage	H3/7.5	+0.33 [−0.97, +1.82]	−8.3
Outage	H3/6	+0.98 [−0.40, +2.59]	−10.0
Outage	H5/12.5	−1.44 [−2.78, +0.07]	−15.6
Outage	H5/15	−1.54 [−3.04, +0.15]	−16.9
Fading	H3/9	−0.01 [−1.90, +1.73]	−5.4
Fading	H3/7.5	−3.82 [−6.04, −1.71]	−21.3
Fading	H3/6	−8.62 [−10.95, −6.39]	−33.8
Fading	H5/12.5	−3.33 [−6.00, −0.72]	−20.2
Fading	H5/15	−2.18 [−5.71, +1.19]	−8.6
Stress	H3/9	−2.97 [−5.25, −0.99]	−16.4
Stress	H3/7.5	−4.51 [−5.45, −3.57]	−28.8
Stress	H3/6	−7.86 [−9.58, −6.28]	−39.7
Stress	H5/12.5	−6.22 [−8.32, −3.58]	−33.2
Stress	H5/15	−5.32 [−7.13, −3.58]	−23.2

5.4 Discussion

What does this mean for someone designing HELLO handling? Mostly that the HELLO rate and the hold time should not be tuned as one knob. With the stock coupling, a cheaper HELLO also means slower purging, and faster purging means more traffic. Pulling the two apart gave something stock OLSR cannot offer. In the churn and outage regimes we tested, keeping the stock hold time while sending fewer HELLOs kept departure detection as fast as stock (0.55 s faster, in fact) and saved bytes. The cost is 1.6 s slower discovery of returning neighbors, which did not show up as a PDR loss in those scenarios.

It also limits what adaptive HELLO schemes can realistically achieve. A node that has left cannot shorten its own detection, because the purge is the receiver's decision. Adaptation that matters has to act on the receiver's expiry, for example by advertising a validity time that reflects the link quality the receiver sees, or by using link-layer failure feedback. The static setting we tested is a first step only, and the fading results show where it stops working: a hold factor near 2 needs a clean channel.

No single static setting won everywhere, so we looked at what a per-scenario choice would give. Table 5 picks, for each scenario, the setting with the largest byte saving among those whose mean PDR is within 0.5 pp of the stock default or better. Where none qualifies, it keeps the stock default. The rule uses point estimates and was written after the results were in, so read the table as description, not as a test.

Table 5 - Best observed setting per scenario, selected from Table 4 (pooled over sizes; PDR and bytes are differences against the stock default H2/6)

Scenario	Setting	PDR difference (pp) [95% CI]	Bytes difference (%)
Stable	H5/15	+1.54 [+0.01, +3.62]	−16.9
Churn	H3/6	+2.71 [+0.35, +5.86]	−10.5
Outage	H3/6	+0.98 [−0.40, +2.59]	−10.0
Fading	H3/9	−0.01 [−1.90, +1.73]	−5.4
Stress	H2/6 (stock)	reference	reference

The pattern is easy to read. A slow HELLO is free when nothing changes and expensive when neighbors come and go. On a reliable channel H3/6 s is the better trade, on fading channels only the factor-of-3 setting H3/9 s keeps the stock delivery ratio, and under stress nothing we tried beats the default. A controller that raised the hold factor when the receiver saw many lost HELLOs, and lowered it on a clean channel, might reproduce much of this table without a per-scenario choice. We have not built one, so that is a guess.

5.5 Limitations and threats to validity

This is a simulation study with one grid family, 802.11b at 1 Mbit/s, five flows, and a single leaver in the trace experiment. Results may differ for other topologies, mobility models (Camp et al., 2002), and radios, and testbed work suggests that simulation can miss losses that appear in real deployments (Ikeda et al., 2009). Control bytes are OLSR packet bytes, not airtime or energy, and the byte savings shrink with network size (17.8% to 4.7% under churn). About 50 paired comparisons were made in Experiment B without correction for multiplicity, so the positive PDR differences are exploratory, and several intervals include zero. The selection rule behind Table 5 uses point estimates and was chosen after seeing the results. The decoupled settings need a one-attribute patch to the stock model. We did not test interoperability with unmodified receivers; receivers that honor the validity-time field should follow the advertised value, but this remains untested.

Experiment C tested one adaptive controller design, so the result does not rule out every adaptive scheme, in particular receiver-side adaptation. Node 0 is isolated in every setting in the outage and stress scenarios, which caps the PDR but does not bias the paired comparison. The staleness and missing-entry counts use evaluator-only ground truth from node positions, and the missing-entry count is not meaningful under fading.

Conclusion

In OLSRv1 a departed neighbor disappears when the validity time of its last HELLO runs out, and a returning one appears after about 1.5 HELLO intervals. Two different timers govern the two events, even though the stock configuration ties them together. The traces are the evidence for that, and it is enough to explain why speeding up the sender did not help in our earlier controller study. Setting HELLO to 3 s with a 6 s hold time saved a modest amount of control traffic in stable, churn, and outage conditions without a visible cost in delivery. It failed on fading channels, and no static setting was best everywhere. What we have not shown is that a receiver-side adaptive hold time would fix this. That is the obvious experiment to run next, along with mobility models, other radios, and receivers that run unmodified OLSR.

Data availability

The following files are provided: son_study2.cc (system-level study), son_trace.cc (trace experiment), olsr_patch_NeighbHoldFactor.txt (patch), run_all.sh and run_trace.sh (drivers), SON_V2_Protocol.txt (protocol written before the runs), son_ns3_v2_results.csv (900 runs), son_trace_results_1.csv (1,440 observer records), and son_ns3_confirmatory.csv (data of Experiment C). Each run is a separate ns-3.49 process with the seed and run number fixed. All artifacts are publicly available in the SON_Freshness repository at https://github.com/DinaHatrak/SON_Freshness.

REFERENCES

- Andel, T. R., & Yasinsac, A. (2006). On the credibility of MANET simulations. *Computer, 39*(7), 48–54.
- Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y.-C., & Jetcheva, J. (1998). A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)* (pp. 85–97).
- Camp, T., Boleng, J., & Davies, V. (2002). A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing, 2*(5), 483–502.
- Clausen, T., Dearlove, C., & Dean, J. (2011). *Mobile ad hoc network (MANET) neighborhood discovery protocol (NHDP)* (RFC 6130). Internet Engineering Task Force.
- Clausen, T., Dearlove, C., Jacquet, P., & Herberg, U. (2014). *The optimized link state routing protocol version 2* (RFC 7181). Internet Engineering Task Force.
- Clausen, T., & Jacquet, P. (Eds.). (2003). *Optimized link state routing protocol (OLSR)* (RFC 3626). Internet Engineering Task Force.
- Clausen, T. H., Hansen, G., Christensen, L., & Behrmann, G. (2001). The optimized link state routing protocol: Evaluation through experiments and simulation. In *Proceedings of the Symposium on Wireless Personal Multimedia Communications (WPMC)*, Aalborg, Denmark.
- Corson, S., & Macker, J. (1999). *Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations* (RFC 2501). Internet Engineering Task Force.
- De Couto, D. S. J., Aguayo, D., Bicket, J., & Morris, R. (2003). A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*.
- Ernst, R., & Martini, P. (2012). Adaptive HELLO for the neighborhood discovery protocol. In *Proceedings of the IEEE Conference on Local Computer Networks (LCN)*. https://doi.org/10.1109/LCN.2012.6423663
- He, D., Mitton, N., & Simplot-Ryl, D. (2013). An energy efficient adaptive HELLO algorithm for mobile ad hoc networks. In *Proceedings of the ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)* (pp. 65–72).
- Hiyama, M., Sakamoto, S., & Kulla, E. (2013). Experimental results of a MANET testbed for different settings of HELLO packets of OLSR protocol. *Journal of Mobile Multimedia, 9*(1–2), 27–38.
- Huang, Y., Bhatti, S. N., & Parker, D. (2006). Tuning OLSR. In *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. https://doi.org/10.1109/PIMRC.2006.254160
- Ikeda, M., Barolli, L., De Marco, G., Yang, T., Durresi, A., & Xhafa, F. (2009). Tools for performance assessment of OLSR protocol. *Mobile Information Systems, 5*(2), 165–176. https://doi.org/10.3233/MIS-2009-0079
- Ingelrest, F., Mitton, N., & Simplot-Ryl, D. (2007). A turnover based adaptive HELLO protocol for mobile ad hoc and sensor networks. In *Proceedings of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*.
- Jacquet, P., Mühlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., & Viennot, L. (2001). Optimized link state routing protocol for ad hoc networks. In *Proceedings of the IEEE International Multi Topic Conference (INMIC)* (pp. 62–68).
- Johnson, D., Hu, Y., & Maltz, D. (2007). *The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4* (RFC 4728). Internet Engineering Task Force.
- Jovel, F. (2014). *An optimization of neighbor discovery in mobile ad hoc networks* [Thesis, University of Texas at El Paso].
- Kurkowski, S., Camp, T., & Colagrosso, M. (2005). MANET simulation studies: The incredibles. *ACM SIGMOBILE Mobile Computing and Communications Review, 9*(4), 50–61. https://doi.org/10.1145/1096166.1096174
- Li, X., Mitton, N., & Simplot-Ryl, D. (2011). Mobility prediction based neighborhood discovery in mobile ad hoc networks. In *NETWORKING 2011* (Lecture Notes in Computer Science, Vol. 6640). Springer.
- Mitton, N., Saleem, Y., Loscri, V., & Bureau, C. (2024). Adaptive HELLO protocol for vehicular networks. *ITU Journal on Future and Evolving Technologies, 5*(1), 147–157. https://doi.org/10.52953/VUYV1591
- Perkins, C., Belding-Royer, E., & Das, S. (2003). *Ad hoc on-demand distance vector (AODV) routing* (RFC 3561). Internet Engineering Task Force.
- Perkins, C. E., & Bhagwat, P. (1994). Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of ACM SIGCOMM* (pp. 234–244).
- Razafindralambo, T., & Mitton, N. (2007). Analysis of the impact of hello protocol parameters over a wireless network self-organization. In *Proceedings of PE-WASUN* (pp. 46–53). Chania, Greece.
- Riley, G. F., & Henderson, T. R. (2010). The ns-3 network simulator. In *Modeling and tools for network simulation* (pp. 15–34). Springer.
- Xu, Z., Li, X., & Xu, D. (2024). Research on cost optimization of UAV network routing protocol based on OLSR protocol. In *6GN for future wireless networks* (Lecture Notes of ICST, Vol. 554, pp. 175–187). Springer. https://doi.org/10.1007/978-3-031-53404-1_15