# D-PLCA Follower Node ID Time Randomization (Comment I-97)


A Leading Provider of Smart, Connected and Secure Embedded Control Solutions


**Tim Baggett**

IEEE 802.3da - September 2025 Interim – Minneapolis, MN

# Acknowledgments

**Patrick Somers (Microchip)**

**David Law (HPE)**

# Background

- **New D-PLCA nodes joining the network wait in the LEARNING state for a number of PLCA cycles defined by the aging_cycles variable.**
  - This time is spent listening to the network building up a table of claimed and unclaimed transmit opportunities.
    - A “claim” means a packet was detected in that transmit opportunity
- **Once the wait is completed, the node immediately selects the smallest unclaimed TO from the claim table in the FOLLOWER state**

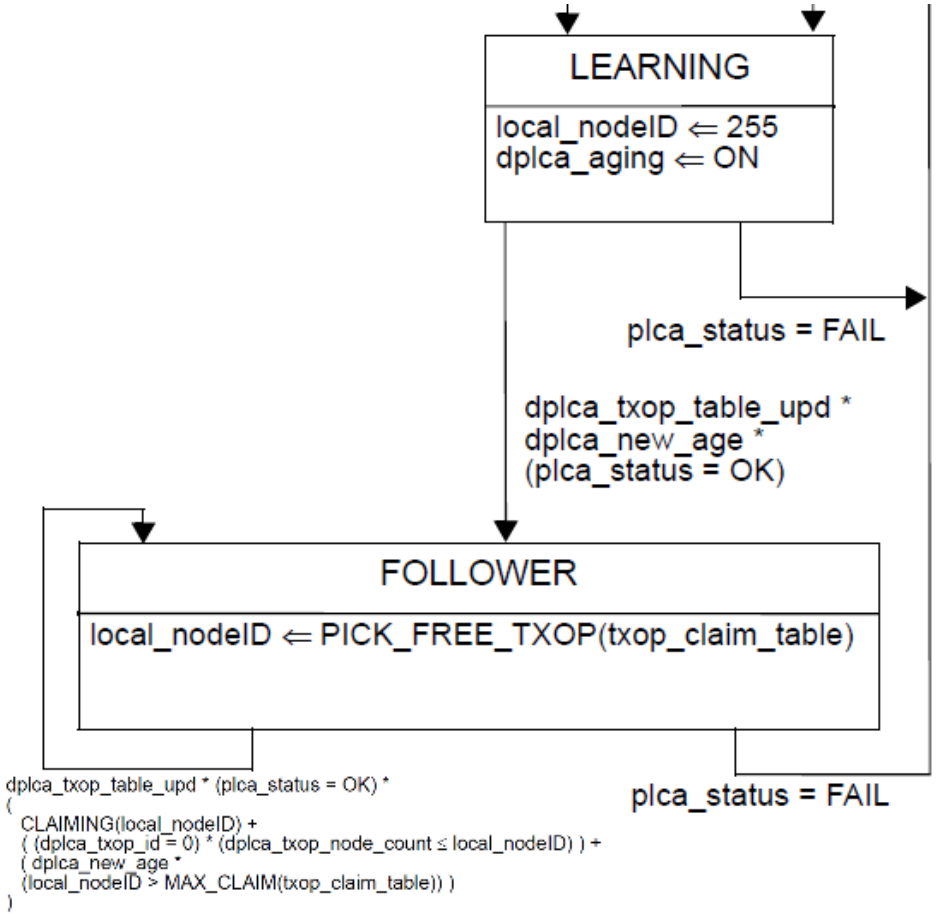



Fig 148-8 – D-PLCA Control State Diagram (pg 82)

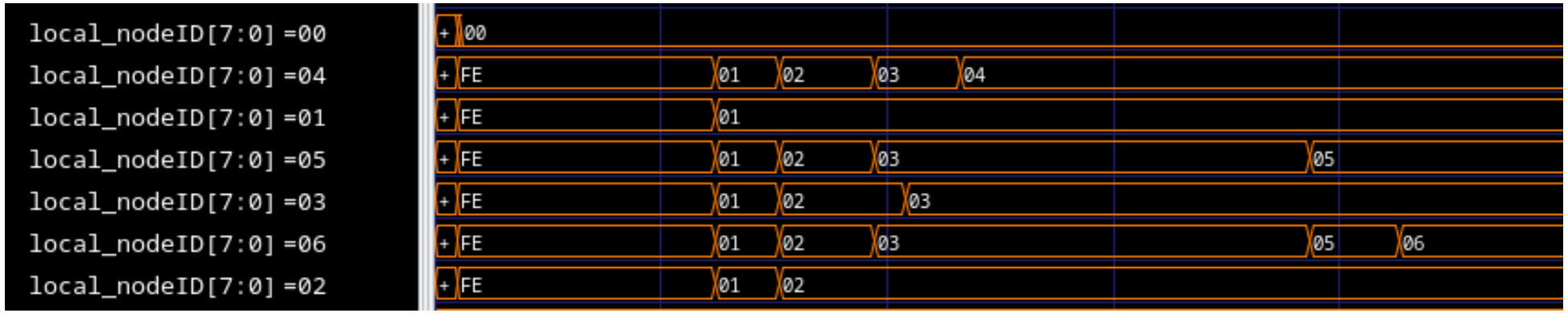
# Problem

- **When Follower nodes first detect a BEACON, they enter the LEARNING state synchronously. They then select the same lowest claimed TO.**
  - The first node to successfully transmit a packet in the TO, wins the claim
  - But – multiple nodes *may* attempt to transmit in the same TO
    - Collisions are expected during start-up and convergence of the D-PLCA algorithm
    - Probability of collisions depends on transmit traffic patterns

- **Consider a segment of identical devices (same hardware, firmware, etc.) yielding identical behavior.**
  - If MPoE is used, they all will get powered-up simultaneously
  - The identical behavior will result in simultaneous transmission resulting in significant collisions and packet loss

# Problem

- Plot below illustrated the convergence of six Follower D-PLCA nodes on start-up.
- At reception of the first BEACON, the Follower nodes enter the LEARNING state and listen.
- Upon entering the FOLLOWER state for the first time all nodes select ‘1’ which one successfully transmits and claims
- Remaining nodes then select TO ‘2’ of which one successfully transmits and claims
- Remaining nodes then all select TO ‘3’, and so on…

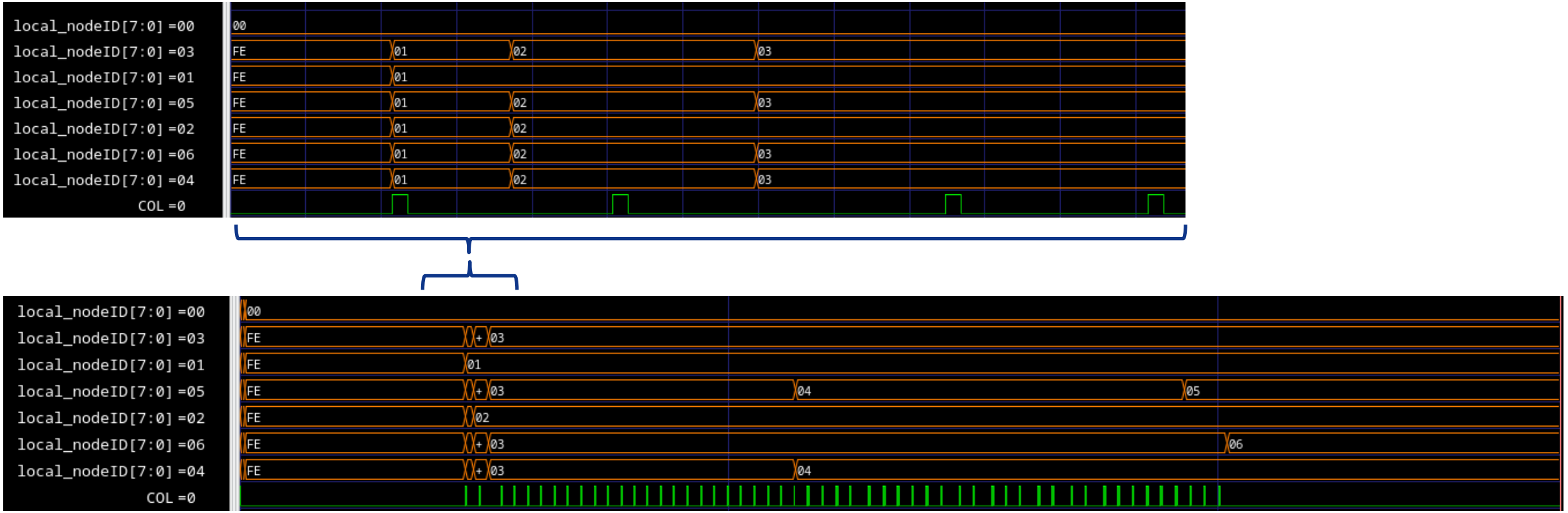


# Simulation of Worst Case

- **Verilog Behavioral model**
  - Segment of 7 nodes: one coordinator, 6 followers
  - Each follower transmits 50 packets
    - No delay between packets, MAC always has a packet pending transmission
  - aging_cycles initially set to 128 for illustrative purposes

# Simulation of Worst Case

- **Simulation Results**
  - 78 collisions (17 packets with excessive collision errors)
  - 0 excessive deferral errors



# Proposed solution

- **The solution proposed is to allow for a random number of PLCA cycles to occur before selecting the lowest unclaimed TO from the claim table.**
  - Nodes that wait less will have more cycles in which to transmit and claim the TO before other nodes that wait longer to pick a TO.
    - Nodes waiting longer will pick a different TO if the earlier node transmits & claims

**Result is that nodes spread out their selection of a TO**

# Fig 148-8 Changes

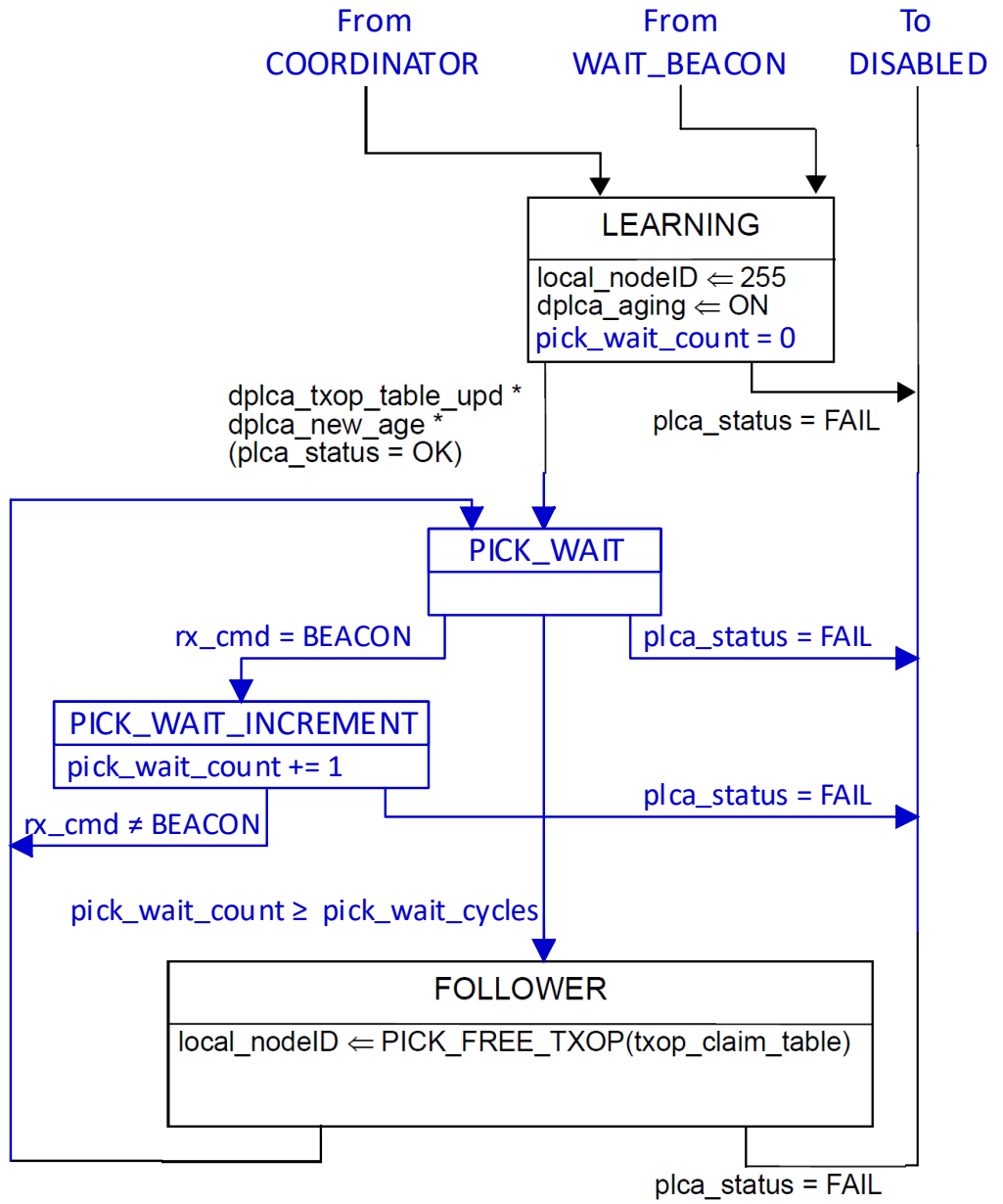
## New D-PLCA Variables 148.4.7.2

- **pick_wait_cycles**

  **This variable is the number of BEACONs that will be received (PLCA cycles) before entering the FOLLOWER state and selecting an unused transit opportunity. The value is a random number selected from the range of 0 to the value of aging_cycles divided by two upon entry into the LEARNING and FOLLOWER states.**

- **pick_wait_count**

  **This variable counts the number of BEACONs received (PLCA cycles) since exiting the LEARNING state**

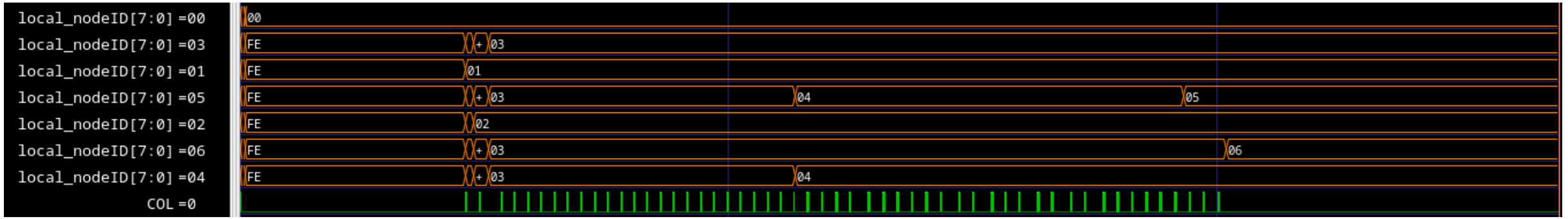



```
dplca_txop_table_upd * (plca_status = OK) *
(
  CLAIMING(local_nodeID) +
  ( (dplca_txop_id = 0) * (dplca_txop_node_count ≤ local_nodeID) ) +
  ( dplca_new_age *
  (local_nodeID > MAX_CLAIM(txop_claim_table)) )
)
```

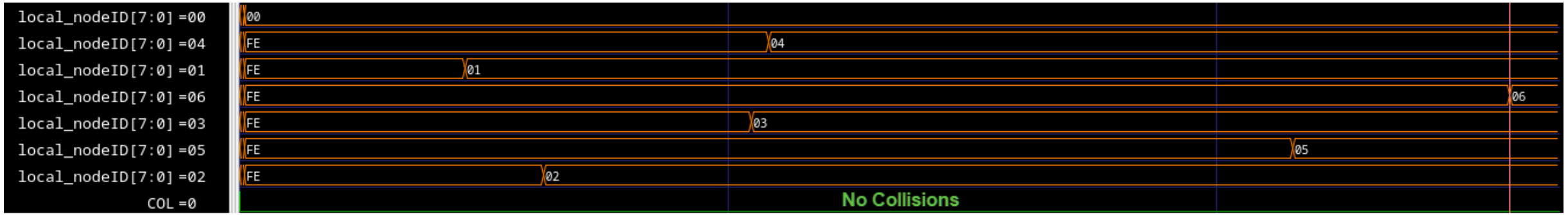
# Simulation of Worst Case with Proposed Change

- **Simulation Results with proposed delay in picking free TO ID**
  - 0 collisions (compare to 78 without the change)
  - 4 excessive deferral errors  (compare to 0 before the change)

**Prior to proposed change:**




**After the proposed change:**



# Simulation of Worst Case with Proposed Change

- **Simulation Results with varying aging_cycles**
  - Each follower transmits 50 packets
  - MAC transmit is always pending

**Significant reduction in start-up collisions for small increase in excessive deferrals and convergence time.**

Note: "Before" proposed change - no delay before picking free TO ID
"After" proposed change - random PLCA cycle delay before picking free TO ID

| aging_cycles | # Collisions | | # Excessive Collisions | | # Excessive Deferrals | | Convergence Time (ms) | |
|---|---|---|---|---|---|---|---|---|
| | Before | After | Before | After | Before | After | Before | After |
| **32** | 78 | 11 | 17 | 1 | 0 | 2 | 16.8 | 6.8 |
| **64** | 77 | 6 | 17 | 0 | 0 | 3 | 17.6 | 12.4 |
| **128** | 78 | 0 | 17 | 0 | 0 | 4 | 20.2 | 26.0 |

# What about random packet transmission?

- **Simulation Results with varying aging_cycles**
  - Each follower transmits 50 packets
  - Uniform random delay between 0-500 µs between packet transmissions

**Start-up collisions reduced traded for convergence time and small increase in excessive deferrals.**

| aging_cycles | # Collisions | | # Excessive Collisions | | # Excessive Deferrals | | Convergence Time (ms) | |
|---|---|---|---|---|---|---|---|---|
| | Before | After | Before | After | Before | After | Before | After |
| **32** | 11 | 13 | 3 | 0 | 0 | 1 | 4.5 | 9.0 |
| **64** | 15 | 6 | 3 | 0 | 0 | 3 | 7.3 | 8.4 |
| **128** | 15 | 0 | 3 | 0 | 0 | 4 | 9 | 19.0 |

# Simulation Results – 50 packets/follower

No delay between follower transmit packets pending

| aging_cycles | # Collisions | | # Excessive Collisions | | # Excessive Deferrals | | Convergence Time (ms) | |
|---|---|---|---|---|---|---|---|---|
| | Before | After | Before | After | Before | After | Before | After |
| **32** | 78 | 11 | 17 | 1 | 0 | 2 | 16.8 | 6.8 |
| **64** | 77 | 6 | 17 | 0 | 0 | 3 | 17.6 | 12.4 |
| **128** | 78 | 0 | 17 | 0 | 0 | 4 | 20.2 | 26.0 |

Random 0-500 µs delay between follower transmit packets pending

| aging_cycles | # Collisions | | # Excessive Collisions | | # Excessive Deferrals | | Convergence Time (ms) | |
|---|---|---|---|---|---|---|---|---|
| | Before | After | Before | After | Before | After | Before | After |
| **32** | 11 | 13 | 3 | 0 | 0 | 1 | 4.5 | 9.0 |
| **64** | 15 | 6 | 3 | 0 | 0 | 3 | 7.3 | 8.4 |
| **128** | 15 | 0 | 3 | 0 | 0 | 4 | 9 | 19.0 |

# Simulation Results – 20 packets/follower

No delay between follower transmit packets pending

| aging_cycles | # Collisions | | # Excessive Collisions | | # Excessive Deferrals | | Convergence Time (ms) | |
|---|---|---|---|---|---|---|---|---|
| | Before | After | Before | After | Before | After | Before | After |
| **32** | 37 | 11 | 9 | 1 | 0 | 2 | 10.6 | 6.8 |
| **64** | 36 | 25 | 9 | 0 | 0 | 4 | 11.4 | 13.4 |
| **128** | 36 | 0 | 9 | 0 | 0 | 4 | 13.6 | 16.2 |

Random 0-1000 µs delay between follower transmit packets pending

| aging_cycles | # Collisions | | # Excessive Collisions | | # Excessive Deferrals | | Convergence Time (ms) | |
|---|---|---|---|---|---|---|---|---|
| | Before | After | Before | After | Before | After | Before | After |
| **32** | 12 | 10 | 0 | 0 | 0 | 1 | 3.1 | 10.0 |
| **64** | 12 | 1 | 0 | 0 | 0 | 3 | 4.2 | 5.9 |
| **128** | 12 | 0 | 0 | 0 | 4 | 4 | 6.5 | 9.8 |

# Conclusion

- **Excessive Deferrals dependent upon the number of aging_cycles**
  - Longer cycle to update the claim table in LEARNING, the longer the packets remain pending
- **Any randomization of time between packet transmission significantly helps**
  - Lowers number of collisions, excessive collisions, and convergence time
- **Delaying a random number of PLCA cycles before picking an unclaimed TO:**
  - Improves performance in described worst-case segment condition
  - Slows down node ID convergence
  - Can increase excessive deferrals
- **Downside could be managed by adjustment of aging_cycles such as using aging_cycles=64 as in the case simulated**

# Conclusions

- **Adding a random wait before picking an unclaimed TO:**
  - Improves performance in described worst-case segment condition
  - Slows down node ID convergence
  - Can increase excessive deferrals

- **Downsides could be managed by adjustment of aging_cycles such as using aging_cycles=64 as in the case simulated**

# Thank You

## Questions?