Canova Tech

The Art of Silicon Sculpting

Dynamic PLCA Node ID Assignment

Piergiorgio Beruto

March, 10th 2021

Outline

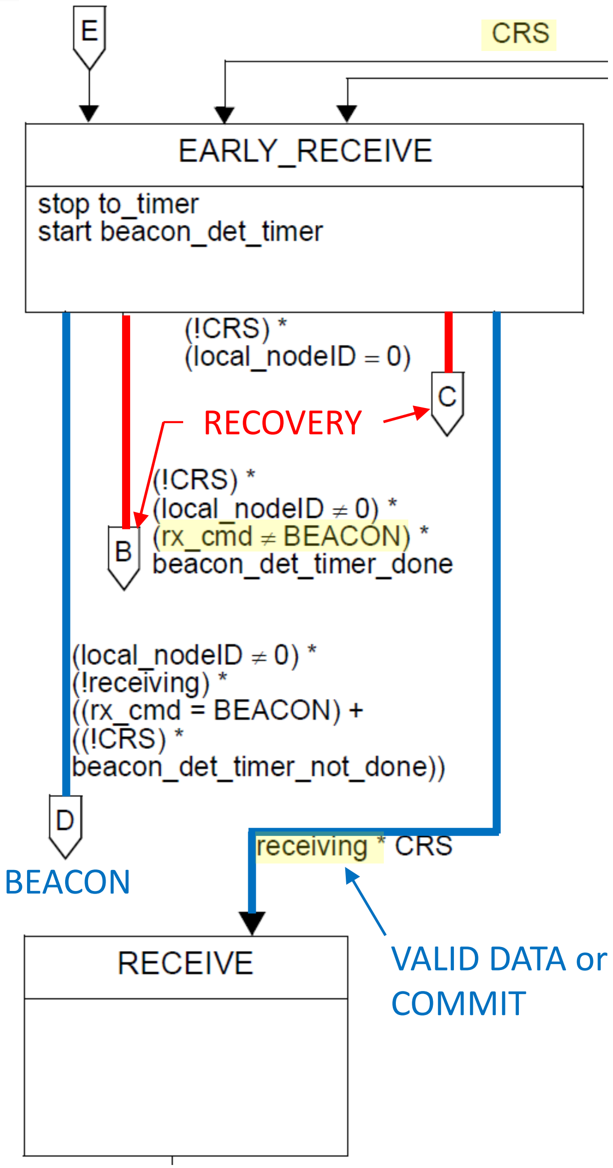
- 802.3da has a formal objective to define an “optional PLCA node ID allocation method”
  - AKA “Dynamic PLCA”, or D-PLCA in short
- Presentations given so far:
  - http://www.ieee802.org/3/SPMD/public/apr0820/spmd_nodeid_040820.pdf
  - https://www.ieee802.org/3/da/public/jul20/jones_spmd_01_0720.pdf
  - https://www.ieee802.org/3/da/public/102120/dalmia_3da_01_102120.pdf
  - https://www.ieee802.org/3/da/public/110420/beruto_3da_01_110420.pdf
  - https://www.ieee802.org/3/da/public/022421/dalmia_3da_022421.pdf
- This presentation proposes a baseline adoption for D-PLCA
  - based on the “physical layer solution” shown in beruto_3da_01_110420.pdf

Goals and requirements

- As discussed in https://www.ieee802.org/3/SPMD/email/msg00223.html
  - Be interoperable with CSMA/CD nodes on the network in a plug-and-play matter, without reconfiguration on detected errors
    - Which also implies: be interoperable with 802.3cg PLCA enabled nodes on the same mixing-segment without specific/additional configuration
  - Be at least as fast as other upper-layer node ID allocation methods (e.g. LLDP)
  - Be compatible with nodes transitioning into a sleep state, where they are powered down and do not communicate
  - keep at least the same level of EMC performance as in 802.3cg
    - a lot of work has been done in 802.3cg not to preclude meeting industrial and automotive requirements
- Additionally
  - Do not hamper current 802.3cg PLCA network performance (latency, throughput, fairness)
  - have D-PLCA be optional to implement /enable (still allowing static PLCA configuration)

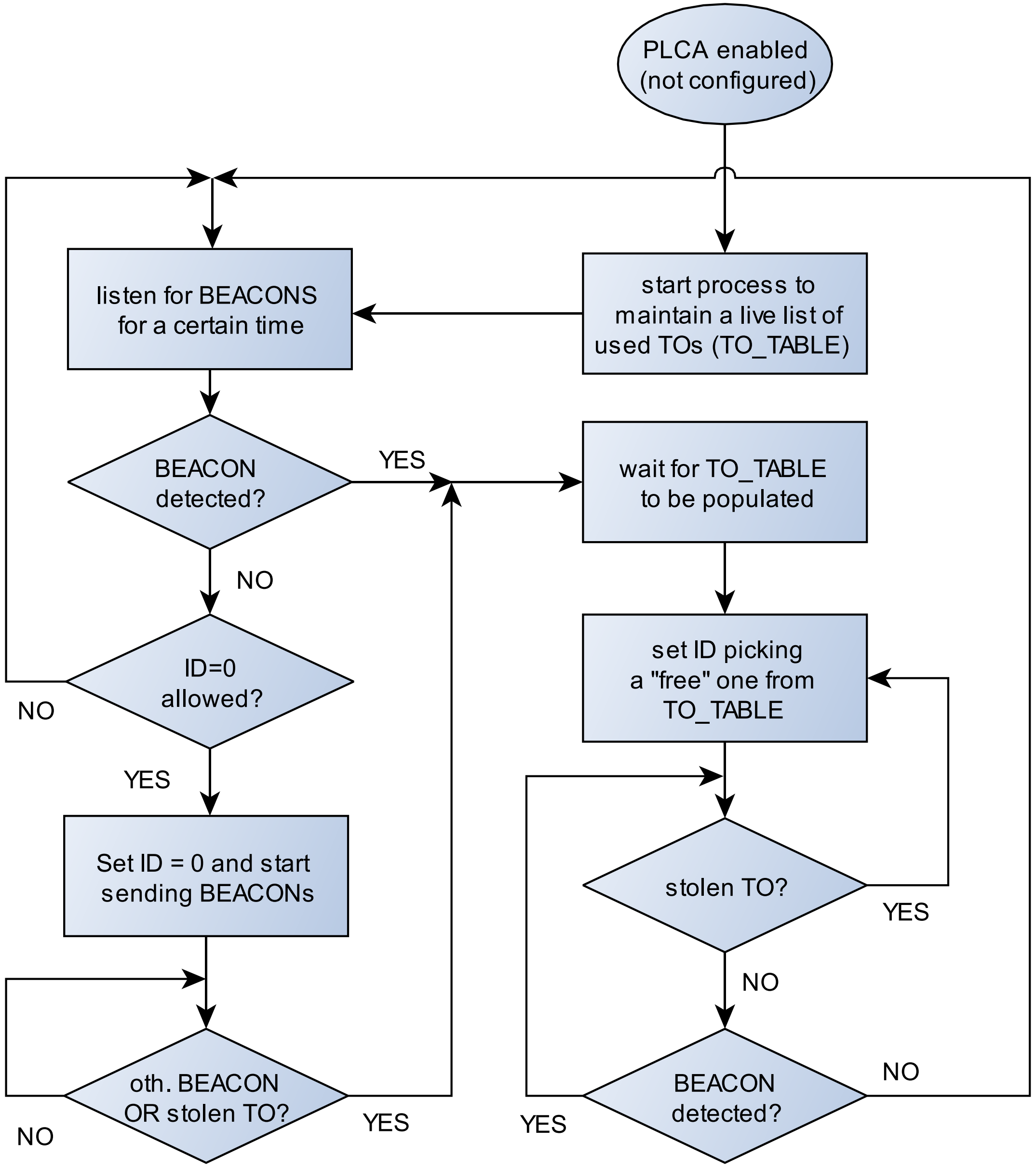
Constraints to preserve 802.3cg compatibility

- We shall not rely on handling detected collisions
  - Collision detection belongs to the Physical Layer but collision **handling** does not
- Do not add any new physical layer signaling
  - Any signal other than a valid preamble, BEACON or COMMIT will be incompatible with Clause 148
  - That would make existing PLCA nodes go into a recovery/resync state →
- We should avoid **periodic** physical layer signaling on the line
  - PLCA nodes would react to that by signaling a collision in case of concurrent transmissions
  - non-PLCA enabled nodes will assert CRS at each transmission, causing deferral and possibly hampering media access fairness
  - may (likely) impact EMC/EMI performance



PROPOSAL

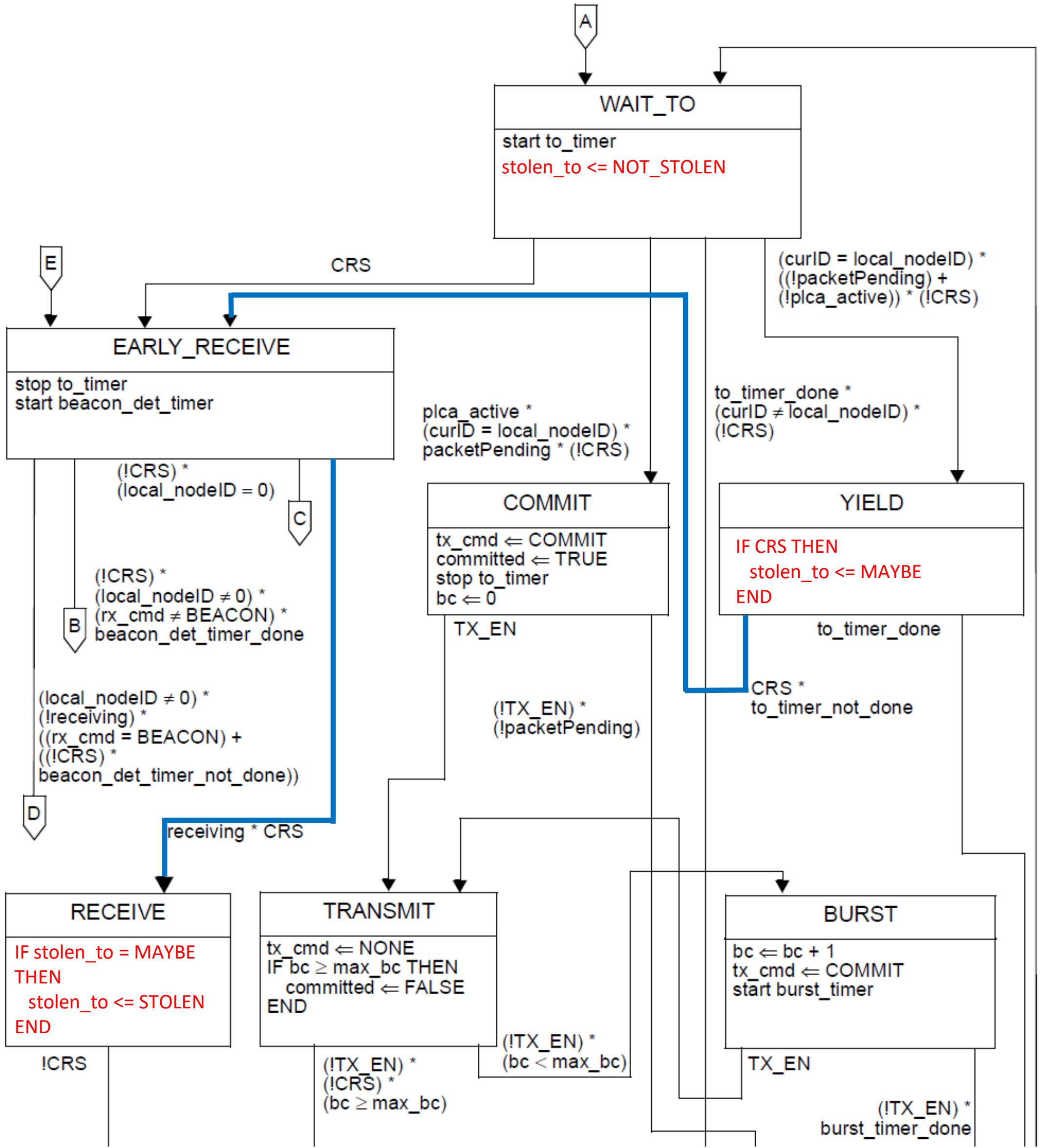
Concept - Overview


PLCA enabled (not configured)
start process to maintain a live list of used TOs (TO_TABLE)
listen for BEACONS for a certain time
BEACON detected?
YES
NO
wait for TO_TABLE to be populated
ID=0 allowed?
NO
YES
set ID picking a "free" one from TO_TABLE
Set ID = 0 and start sending BEACONs
stolen TO?
YES
NO
oth. BEACON OR stolen TO?
NO
YES
BEACON detected?
YES
NO

- Use the "duck" algorithm
  - "If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck"
  - Start over if it wasn't
- Use the concept of "stolen TO"
  - detecting that some other node is transmitting during a node's TO
  - Kind of playing "bocce" where successful transmissions kick other nodes out of the current ID
- Keep a list of used TOs by detecting transmissions
  - free up TOs using an AGING criteria
  - Coordinator (ID = 0) dynamically adjusts plcaNodeCount
- Concept similar to MAC address learning in switches or dynamic Wi-Fi channel selection

Concept - Stolen TO

- We call a stolen TO the event when a node detects another node transmitting during the former's transmit opportunity
- This can happen in the following situations
  - Two or more nodes have the same ID
  - There are non-PLCA nodes on the mixing-segment
- Can be easily detected extracting the information from the **existing** Clause 148 Control State Diagram



Concept - Aging (1)

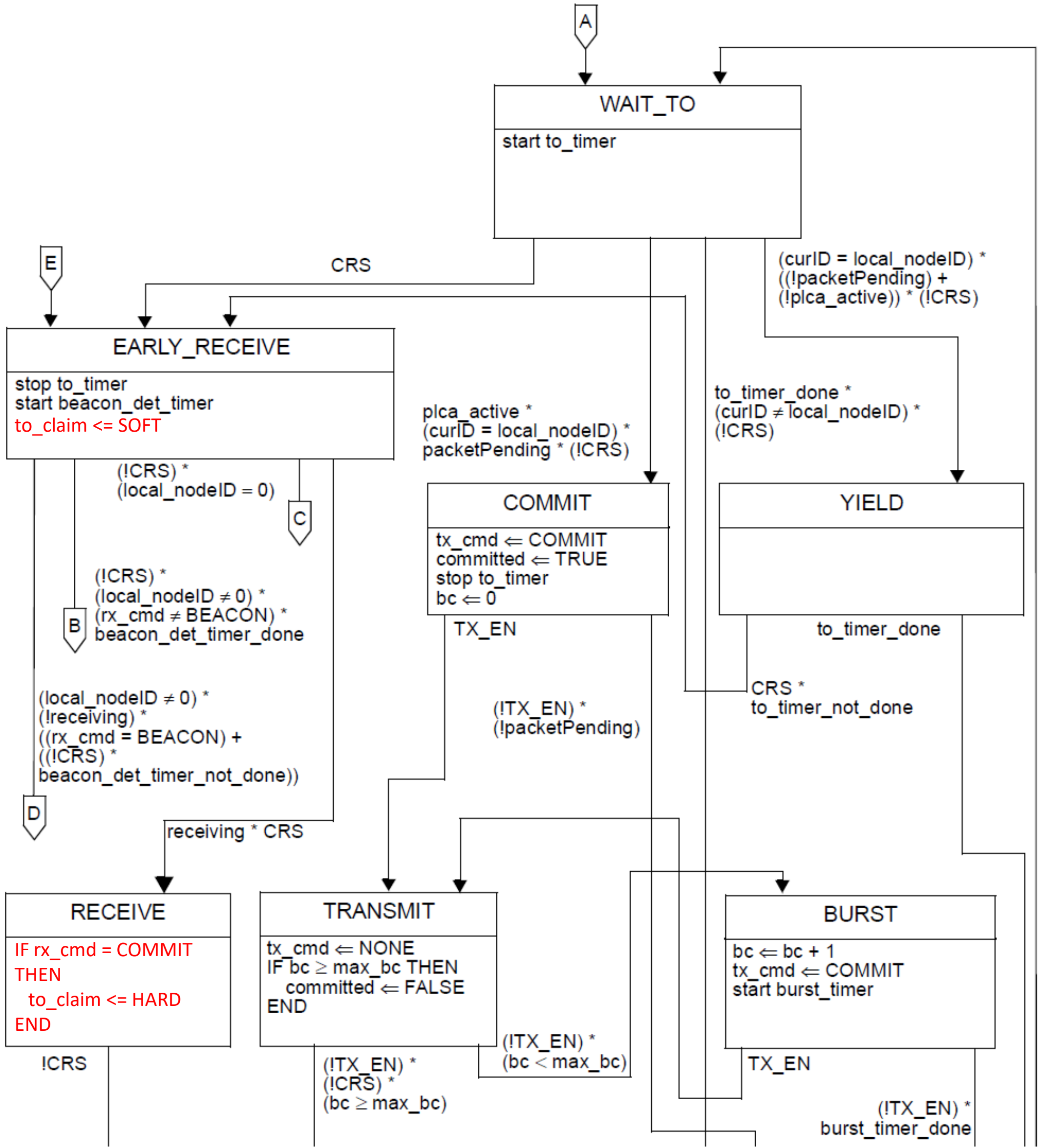
- We want to keep a table of "used" TOs by detecting transmissions
  - allows the follower nodes to select a (possibly) free ID
  - allows the coordinator node (the one sending BEACONs) to adjust plcaNodeCount according to the effective number of nodes on the mixing-segment
  - **every node on the mixing-segment maintains its own table (!)**
- If a node is not using its TO for some time (aging time), it is considered to have left the network, therefore the TO shall be freed
- We want to distinguish at least two different cases of TO claim, with possibly <u>different aging times</u>
  - HARD claim → a COMMIT was detected during the TO
  - SOFT claim → a packet **not** preceded or followed by a COMMIT was detected

Concept - Aging (2)

- Why HARD and SOFT claims?
  - non-PLCA nodes may send packets (w/o COMMIT) at any time, regardless of PLCA TOs
  - 802.3cg PLCA nodes send packets which are **occasionally** preceded by COMMIT
  - In this proposal (see next slides), D-PLCA nodes always send a COMMIT along with a packet → **always do "HARD" TO claims**
  - we don't want to have non-PLCA nodes preventing D-PLCA from converging
  - setting a (very) short aging time for SOFT claims makes D-PLCA nodes eventually re-use the IDs that were temporarily claimed by non-PLCA nodes, without growing the PLCA cycle indefinitely (non-PLCA transmissions are unrelated to PLCA TOs !)
  - At the same time, SOFT occupation allows a much faster convergence when mixing D-PLCA and 802.3cg PLCA nodes
- Alternatively, we could have D-PLCA nodes retain the ID when SOFT-stolen, for a while
  - This makes D-PLCA converge better in presence of non-PLCA nodes but worse in presence of 802.3cg nodes
  - Which is better? → to be discussed in the group

Detecting HARD and SOFT TO claims

- It is possible to extract this information as well from the existing Clause 148 PLCA Control State Diagram



A

WAIT_TO

start to_timer

CRS

(curID = local_nodeID) *
((!packetPending) +
(!plca_active)) * (!CRS)

E

EARLY_RECEIVE

stop to_timer
start beacon_det_timer
to_claim <= SOFT

plca_active *
(curID = local_nodeID) *
packetPending * (!CRS)

to_timer_done *
(curID ≠ local_nodeID) *
(!CRS)

(!CRS) *
(local_nodeID = 0)

C

COMMIT

tx_cmd ⇐ COMMIT
committed ⇐ TRUE
stop to_timer
bc ⇐ 0

YIELD

(!CRS) *
(local_nodeID ≠ 0) *
(rx_cmd ≠ BEACON) *
beacon_det_timer_done

B

TX_EN

to_timer_done

CRS *
to_timer_not_done

(local_nodeID ≠ 0) *
(!receiving) *
((rx_cmd = BEACON) +
((!CRS) *
beacon_det_timer_not_done))

(!TX_EN) *
(!packetPending)

D

receiving * CRS

RECEIVE

IF rx_cmd = COMMIT
THEN
  to_claim <= HARD
END

TRANSMIT

```
tx_cmd ⇐ NONE
IF bc ≥ max_bc THEN
  committed ⇐ FALSE
END
```

BURST

bc ⇐ bc + 1
tx_cmd ⇐ COMMIT
start burst_timer

!CRS

(!TX_EN) *
(!CRS) *
(bc ≥ max_bc)

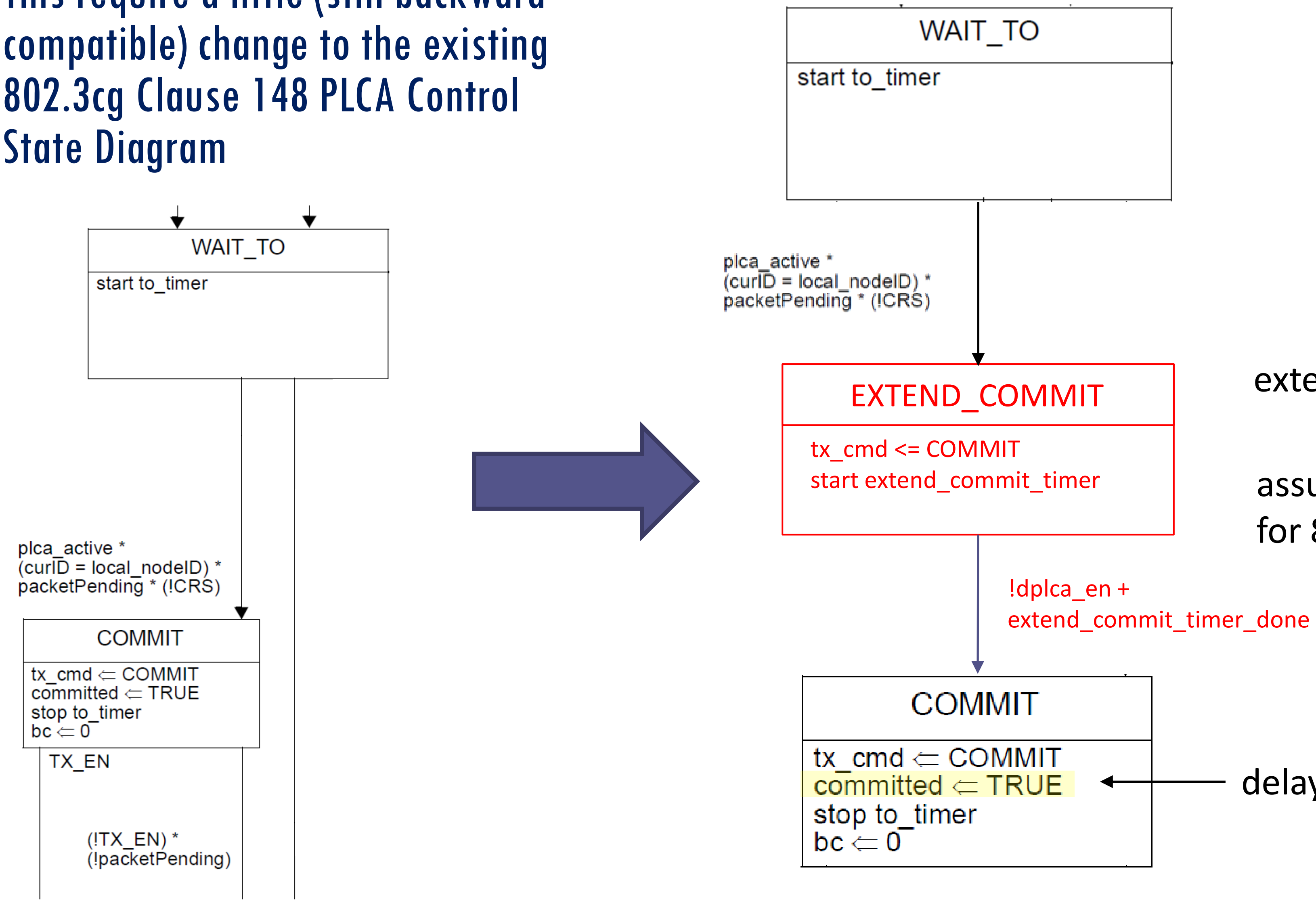
(!TX_EN) *
(bc < max_bc)

TX_EN

(!TX_EN) *
burst_timer_done

Forcing COMMITs when D-PLCA is enabled

- This require a little (still backward compatible) change to the existing 802.3cg Clause 148 PLCA Control State Diagram



extend_commit_timer = 20 BT

assuming dplca_en = FALSE
for 802.3cg nodes

delaying this does the trick!

Mixing “cg” and “da” nodes

- What happens when mixing 802.3cg-compatible nodes with D-PLCA capable nodes?
- Case #1: plugging a D-PLCA node to an existing “cg” network (i.e. non D-PLCA capable coordinator)
  - The D-PLCA node eventually works out a unique ID, avoiding the static (SOFT-occupied) TOs of the 802.3cg nodes
    - eventually, the SOFT-occupation will turn into an HARD-occupation as “cg” nodes will send a COMMIT sooner or later
  - In the meantime, there may be collisions which are detected and handled by the MAC as normal
  - There may be no free TOs to take (i.e. the coordinator’s plcaNodeCount is equal to the actual number of nodes already)
    - In this case, the D-PLCA node won’t be able to achieve enumeration and will keep working in plain CSMA/CD mode creating random collisions.
      - This is what happens already if you plug a non-PLCA node to a PLCA network.
    - If the network load is very low, the D-PLCA node may occasionally steal TOs from non D-PLCA nodes (not a problem…)
  - 👉 In no case the D-PLCA node can prevent a PLCA or non-PLCA node from transmitting, and vice-versa
- Case #2: plugging a “cg” node to a network having a D-PLCA capable coordinator
  - The coordinator will adapt to the highest ID configured in the “cg” nodes
  - eventually, all D-PLCA nodes will detect the “cg” node presence by receiving packets and COMMITs
  - The “cg” node will never release its ID (statically configured), but the D-PLCA nodes do!
- In short: the “cg” nodes win, the D-PLCA nodes adapt to them

EXAMPLES

Digging out the details - Followers enumeration

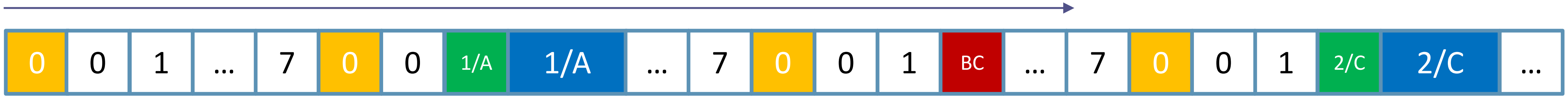
- Have D-PLCA enabled nodes always convey a COMMIT at the beginning or at the end of any transmitted packet
- Have each node monitor the PLCA cycle continuously to collect a list of “occupied” TOs
  - Pick a random “free” TO and set localNodeID accordingly
    - do not pick ID zero (reserved for coordinator)
    - do not pick the last available ID (used for increasing the cycle), unless it is the only one free
  - If at any time a node detects a packet/COMMIT within its own TO, it shall relinquish the current ID and pick a new one
  - Mark a TO as “free” if no packets/COMMITs are received within the aging time
  - The node with the highest ID shall dynamically “move” to a lower ID when possible (after the aging time)
- NOTE: collisions are detected as normal and handled by the MAC
  - CSMA/CD random back-offs resolve conflicts
  - even in the (very) unlikely case of undetected collision, there always will be a new stolen TO eventually

Digging out the details - Followers enumeration

- The coordinator node shall also dynamically adapt the plcaNodeCount parameter to the number of nodes detected
  - The plcaNodeCount sets the number of transmit opportunities between two BEACONs
  - Always keep at least one TO free at the end (plcaNodeCount > highest ID detected), increasing plcaNodeCount accordingly
  - Decrease the plcaNodeCount if no node is claiming the TO before last (down to a minimum of 8)

Example: join of nodes

3 nodes (A, B, C) want to join, coordinator already selected, initial plcaNodeCount = 8



0	0	1	…	7	0	0	1/A	1/A	…	7	0	0	1	BC	…	7	0	0	1	2/C	2/C	…

nodes wait for one cycle
then all take ID = 1

duck hunting!

B and C hear COMMIT
from A and take ID=2

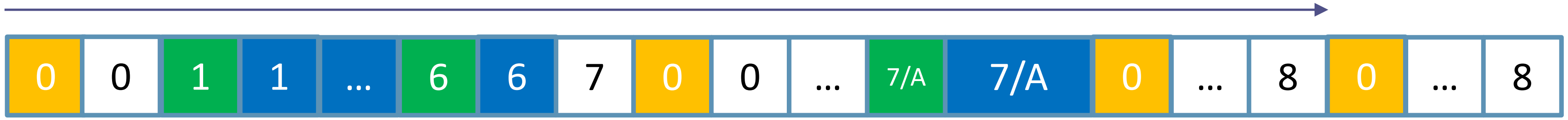
duck!

B and C may collide as they
still share the same ID

did not duck!

B hears COMMIT from C
and takes ID=3

2 nodes (A, B) want to join, currently 7 on the network, initial plcaNodeCount = 8



0	0	1	1	…	6	6	7	0	0	…	7/A	7/A	0	…	8	0	…	8

nodes wait for one cycle
then both choose ID=7
as it’s the only one free

B can’t take ID=8 yet as all
TOs are taken.
But the coordinator increases
plcaNodeCount to 9

B takes ID=8

🟨 BEACON

🟥 COLLISION

🟩 COMMIT

🟦 DATA

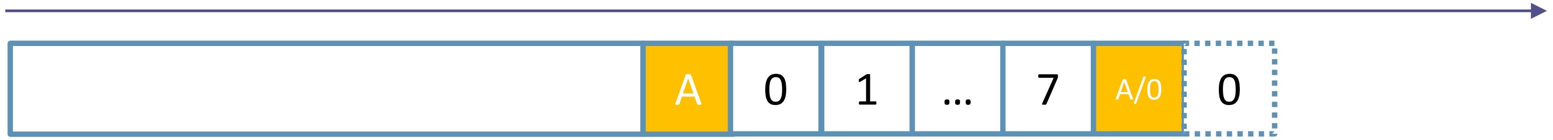
⬜ SILENCE

Digging out the details - Coordinator selection

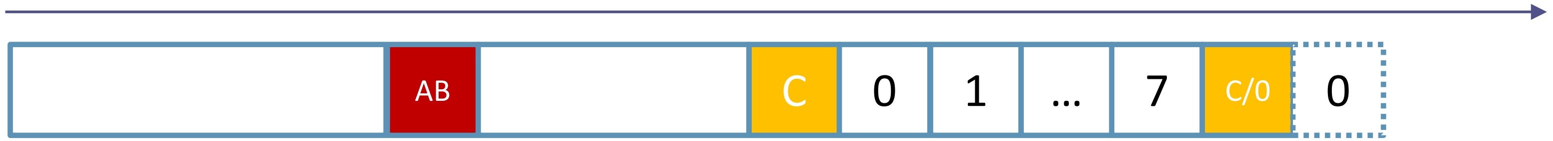
- Nodes that are eligible (configured) for getting the coordinator role shall constantly monitor the line for BEACONs
- if no BEACONs are detected within some time, set localNodeID = 0 and take the coordinator role
- if the coordinator detects a BEACON from another node, or it detects a COMMIT issued by another node within TO #0 → relinquish the coordinator role and go for normal enumeration
  - **note** that this doesn't involve detecting collisions
- Eventually, only one coordinator is selected by statistical convergence
  - Multiple BEACONs on the same mixing-segment temporarily affect performance/fairness but they don’t prevent transmissions
- Why detecting only stolen TOs that include a COMMIT?
  - as said, normal CSMA/CD nodes w/o PLCA may transmit at any time, including during the coordinator’s TO
  - we don’t want such nodes to kick the current coordinator out of its role
  - non PLCA nodes cannot send COMMITs by definition, therefore we can use this information to ignore TOs stolen this way

Example: election of coordinator (localNodeID = 0)

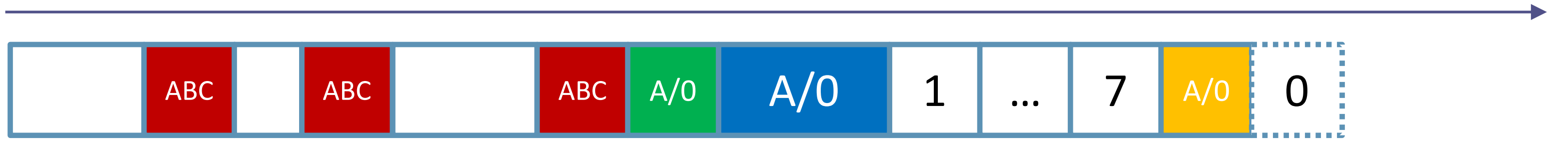
3 nodes (A, B, C) eligible to take the coordinator role, plcaNodeCount = 8 (default)



Ex 1: simple case, node A sends the BEACON first, nodes B and C “hear” it and renounce



Ex 2: BEACONs from A and B collide, then A and B hear the BEACON from C and both renounce.



Ex. 3: worst case, BEACONs from A, B, C collide repeatedly then nodes B and C detect the COMMIT from A and renounce.
If the packet from A collided, then **the MAC** would re-transmit after the usual random back-off

🟨 BEACON

🟥 COLLISION

🟩 COMMIT

🟦 DATA

□ SILENCE

NOTE that during this time nodes can still send/receive data in plain CSMA/CD mode

CONCLUSIONS

Summary

- A method (D-PLCA) for dynamically assign PLCA IDs within the physical layer was presented
  - complies with the current definition of PLCA in 802.3cg
  - meets the goals and requirements discussed in the group
  - allows seamless interoperability with 802.3cg nodes and non-PLCA nodes
- Work to be done
  - Define the aging criteria, evaluating the trade-offs
    - decide whether to tune performance towards interop with 802.3cg nodes or co-existence with non-PLCA nodes
  - Translate this into new state diagram(s) in Clause 148

THANK YOU