

Enhanced PLCA (EPLCA) – Technical Feasibility Study For Predictable Max Latency – Study2

Subramaniam Aravindhan (Arvind)

Synopsys, Inc.

Version 1.3 – Oct 1st, 2026

IEEE 802.3 PLCA Enhancements Study Group

Feasibility Study Options

- Enhancement to Markus Jochim's "Modulo Operation based Diff Serv" Proposal
- EPLCA Proposal
 - Predictable Latency Cyclic-Traffic and Best-Effort Traffic Classes
 - Mechanism to Skip Best-Effort Traffic Class in a PLCA Cycle
 - Plug and Play Discovery/Scheduling Support
 - Backward compatibility to PLCA
 - Bandwidth Utilization Based Latency Improvements – ~25-35%

Enhancement to Markus Jochim's “Modulo Operation based Diff Serv”

- In Markus' proposal, the PLCA Cycle will still need Max Number of Nodes TOs with some of them being empty TO
 - If there are 16 Nodes and only 4 are doing transfer in a PLCA cycle, the number of TO's in a PLCA cycle is still 16, 12 are empty TOs.
- This enhancement is to avoid the empty TOs and have a tighter PLCA cycle for better bandwidth utilization
- Change-1:
 - Keep the Hyper PLCA cycle fixed at 16 (likely we don't need more granularity), 0-15 PLCA cycles in a Hyper PLCA cycle.
 - A Node indicated it's PLCA Interval (wants to do transfer on every PLCA cycle or every other, etc..) (Interval 1-16)
 - The static scheduling algorithm will create a scheduling table for each node. The following information are generated:
 - The scheduling algorithm spits out the Maximum TO's in a PLCA cycle (TOCount information for the coordinator)
 - Each Node will have 16-bit TO-Enabled register, on bit indicating whether transfer is enabled in a PLCA cycle (TOEn[15:0])
 - Each Node will have 16, 8-bit registers indicating PLCA TO-ID (TOID[16][7:0])
 - In a PLCA cycle, if Transfer-enabled is “1” (TOEn[PLCA-cycle]=1), the Node will wait to for its TOID[PLCA-cycle] TO to start the transfer.
 - Changing from NodeID and NodeCount concept to TOCount, TOEn, and TOID for scheduling
 - Scheduling Recommendation: Keep the smaller PLCAInterval traffic at lower TO's and fit larger PLCAInterval traffic at higher TOs
 - This is still static scheduling an no Dynamic TO optimization
- Existing PLCA device Interop:
 - The PLCA devices NodeIDs are assigned from 1 and change TOCount to TOCount+Num-of-PLCA-Devices+1. Shift the EPLCA scheduling table by Num-of-PLCA-Devices.

Enhancement to Markus Jochim's "Modulo Operation based Diff Serv"

Example: Scheduling Table: 16 Nodes, but only 4 TO's in a PLCA Cycle (TOCount=4)

<----- One Hyper PLCA Cycle ----->																		
PLCA Interval	NodeID V	PLCA ->	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	TOEn[15:0]	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
		TOID[16][7:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	TOEn[15:0]	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
		TOID[16][7:0]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	TOEn[15:0]	X		X		X		X		X		X		X		X	
		TOID[16][7:0]	2		2		2		2		2		2		2		2	
2	3	TOEn[15:0]		X		X		X		X		X		X		X		X
		TOID[16][7:0]		2		2		2		2		2		2		2		2
4	4	TOEn[15:0]	X				X				X				X			
		TOID[16][7:0]	3				3				3				3			
4	5	TOEn[15:0]		X				X				X				X		
		TOID[16][7:0]		3				3				3				3		
4	6	TOEn[15:0]			X				X				X				X	
		TOID[16][7:0]			3				3				3				3	
4	7	TOEn[15:0]				X				X				X				X
		TOID[16][7:0]				3				3				3				3
8	8	TOEn[15:0]	X								X							
		TOID[16][7:0]	4								4							
8	9	TOEn[15:0]		X								X						
		TOID[16][7:0]		4								4						
8	10	TOEn[15:0]			X								X					
		TOID[16][7:0]			4								4					
8	11	TOEn[15:0]				X								X				
		TOID[16][7:0]				4								4				
8	12	TOEn[15:0]					X								X			
		TOID[16][7:0]					4								4			
8	13	TOEn[15:0]						X								X	No	
		TOID[16][7:0]						4								4		
8	14	TOEn[15:0]							X								X	
		TOID[16][7:0]							4								4	
8	15	TOEn[15:0]								X								X
		TOID[16][7:0]								4								4

More Nodes
|
V

Enhancement to Markus Jochim's “Modulo Operation based Diff Serv”

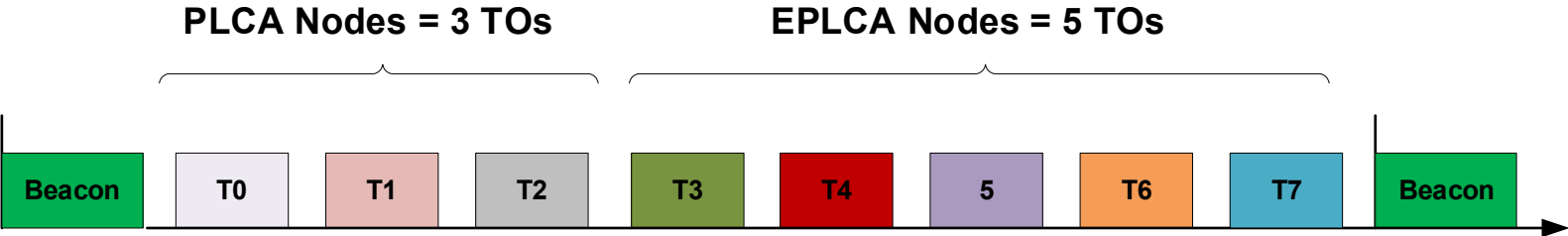
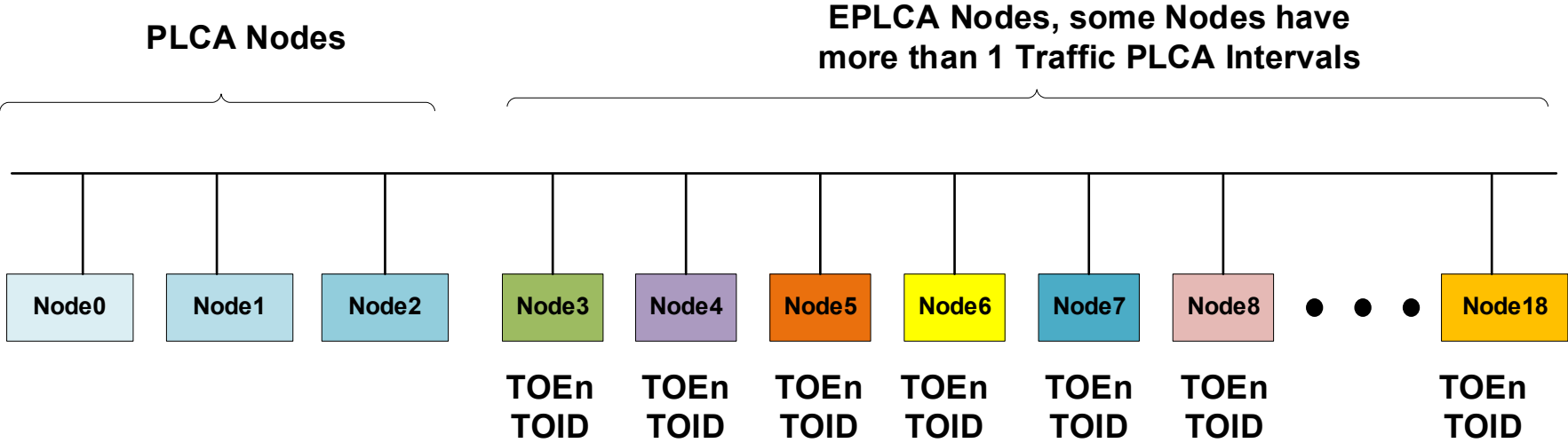
- Change-2: Same concept of PCP/multi-PLCA ID, different implementation.
 - Instead of just one PLCA Interval traffic we could have multiple PLCA Interval traffics for each Node, example up to 3 traffic intervals (A, B, C)
 - Example: PLCA Interval 1 (Low Latency), PLCA Interval 4 (Intermediate Latency), and PLCA Interval 16 (High Latency/Best-Effort)
 - This allows a Node which has low-latency and non-low-latency traffic to use different TO's in a PLCA cycle.
 - Example: Helps a Node to send a Periodic-Traffic at a predictable interval (audio data) and also send Best-Effort traffic once in a while without having to share the same TO.
 - Instead of having one set up TOEn, TOID we will have 3 sets (traffic classes); one for each traffic interval.
Example:
 - (TOEnA, TOIDA), (TOEnB, TOIDB), and (TOEnC, TOIDC), in the future can be expanded up to 16.
 - A node could have same PLCA Interval for all traffic classes A, B, & C. Example: It has 3 critical low-latency sensors
 - Provides more flexibility and encourages fair traffic class usage
- Further Enhancement:
 - We could have “CoordinatorRoleAllowed” feature of the D-PLCA; allows smooth recovery when a coordinator fails. The TOEnx/TOIDx still remains the same and it does not change scheduling; only controls who manages Beacon and PLCA cycle

Enhancement to Markus Jochim's "Modulo Operation based Diff Serv" idea

Example: Scheduling Table: 16 Nodes, some Nodes have more than 1 traffic Intervals; TOCount=5

One Hyper PLCA Cycle																		
PLCA Intervals	NodeID V	PLCA Cycle->	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A=1	0	TOEnA[15:0]	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
		TOIDA[16][7:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B=8	0	TOEnB[15:0]	X								X							
		TOIDB[16][7:0]	5								5							
C=16	0	TOEnC[15:0]		X														
		TOIDC[16][7:0]		5														
A=1, B=0, C=0	1	TOEnA[15:0]	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
		TOIDA[16][7:0]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
A=2, B=0, C=0	2	TOEnA[15:0]	X		X		X		X		X		X		X		X	
		TOIDA[16][7:0]	2		2		2		2		2		2		2		2	
A=2, B=0, C=0	3	TOEnA[15:0]		X		X		X		X		X		X		X		X
		TOIDA[16][7:0]		2		2		2		2		2		2		2		2
A=4, B=0, C=0	4	TOEnA[15:0]	X				X				X				X			
		TOIDA[16][7:0]	3				3				3				3			
A=4, B=0, C=0	5	TOEnA[15:0]		X				X				X				X		
		TOIDA[16][7:0]		3				3				3				3		
A=4	6	TOEnA[15:0]			X				X				X				X	
		TOIDA[16][7:0]			3				3				3				3	
B=16, C=0	6	TOEnB[15:0]				X												
		TOIDB[16][7:0]				5												
A=4, B=0, C=0	7	TOEnA[15:0]				X				X				X				X
		TOIDA[16][7:0]				3				3				3				3
A=8, B=0, C=0	8	TOEnA[15:0]	X								X							
		TOIDA[16][7:0]	4								4							
A=8, B=0, C=0	9	TOEnA[15:0]		X								X						
		TOIDA[16][7:0]		4								4						
A=8, B=0, C=0	10	TOEnA[15:0]			X								X					
		TOIDA[16][7:0]			4								4					
A=8	11	TOEnA[15:0]				X								X				
		TOIDA[16][7:0]				4								4				
B=16	11	TOEnA[15:0]					X											
		TOIDA[16][7:0]					5											
C=16	11	TOEnA[15:0]						X										
		TOIDA[16][7:0]						5										
A=8, B=0, C=0	12	TOEnA[15:0]					X								X			
		TOIDA[16][7:0]					4								4			
A=8, B=0, C=0	13	TOEnA[15:0]						X								X	No	
		TOIDA[16][7:0]						4								4		
A=8, B=0, C=0	14	TOEnA[15:0]							X								X	
		TOIDA[16][7:0]							4								4	
A=8, B=0, C=0	15	TOEnA[15:0]								X								X
		TOIDA[16][7:0]									4							4
More Nodes V																		

EPLCA + PLCA Interop – Multiple PLCA Interval Traffics



$$\text{TOCount} = \text{EPLCA ToCount} + 3 = 8$$

EPLCA Proposal

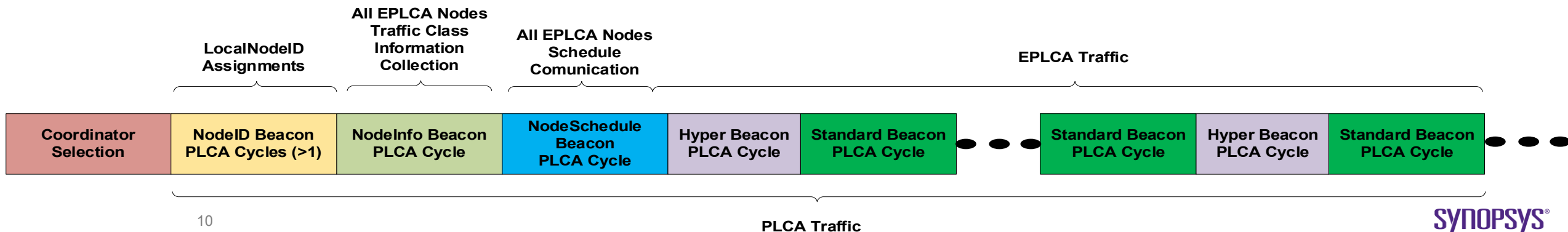
- One Hyper-Cycle consists of 16-PLCA Cycles
- Cyclic-Traffic Class: Traffic-Class with PLCA Interval 1, 2, 4, 8 are Cyclic-Traffic (Do we need 3,5,6,7,9-15 granularity support?) – Latency is predictable
- Best-Effort Traffic Class: Traffic-Class with PLCA Interval 16 are Best-Effort Traffic
- A Node can have up to 3 Traffic Classes: A, B, C (could be same or different PLCA Intervals) - This is scalable up to 16 in future
 - Multi-Node support in a single Device (PLCA Interval=0, no traffic for that class)
- Best-Effort Traffic are scheduled after Cyclic-Traffic in a PLCA Cycle
- Mechanism to indicate SKIP Best-Effort Traffic in a PLCA cycle by Low-latency Cyclic-Traffic-Class Nodes; without wasted bandwidth
 - Provides scalability for high node count applications: 32 Best-Effort Nodes need only 2 TOs.
 - Latency Predictability: Allows up to 8 Cyclic-Traffic Nodes with maximum packet size of 300 bytes (including header) to complete within 2ms PLCA cycle (~2400-byte, 2ms PLCA cycle), plus additional Best-Effort Nodes.
 - This is equivalent to 24 Cyclic-Traffic Nodes each with 103-byte (64+39) payload in a 2ms PLCA cycle; plus additional Best-Effort Nodes.
- Provide Plug and Play Support
- Allow PLCA backward compatibility

Plug and Play Support – Main Flow

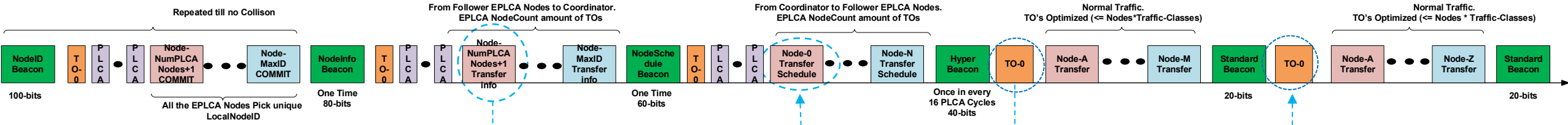
- Static scheduling requires entering scheduling table for all the Nodes one time
 - Cumbersome when there are multiple devices and traffic intervals
 - Need to redo when configuration changes or when new Nodes are added
- Enhancement to support Plug and Play
 - Each Node knows its requirements for traffic (A, B, C: PLCA Intervals and Transfer size in bytes for each Traffic-Class)
 - 1. One Node becomes coordinator – Current D-PLCA mechanism (CoordinatorRoleAllowed)
 - 2. Nodes select unique LocalNodeID during “NodeID Beacon” PLCA cycles (iterative process)
 - 3. Coordinator collects all the Nodes scheduling requirements information during “NodeInfo Beacon” PLCA cycle
 - 4. Coordinator creates the Scheduling Table for all the Nodes/Traffic Class and also TOCount
 - (TOEnA, TOIDA), (TOEnB, TOIDB), and (TOEnC, TOIDC)
 - Coordinator adds two extra TO's
 - End of Cyclic-Traffic TO (;mechanism to skip Best-Effort Traffic)
 - TO-0 is unused TO (;mechanism for a Node to join later)
 - This can be done by firmware/hardware layer; may have to have max limit (example 10ms)

Plug and Play Support – Main Flow

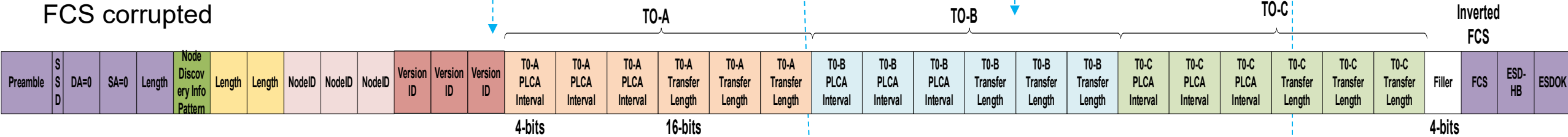
- 5. Coordinator communicates the Scheduling Table to the Nodes during “NodeSchedule Beacon” PLCA cycle – If step 4 takes time, there will be multiple dummy NodeSchedule PLCA cycles.
- 6. Coordinator sends “Hyper Beacon” once in every 16 PLCA Cycles
- 7a. EPLCA Nodes do Ethernet traffic during Hyper and Standard Beacon Cycles.
- 7b. PLCA follower Nodes are added at Lower Node IDs (from ID-1 to ID- NumPLCANodes) for backward compatibility
 - NumPLCANodes Variable defines how many PLCA nodes are present, and TOs are reserved for them (NumPLCANodes=0, when no PLCA backward compatibility Nodes are present)
 - PLCA devices do traffic in all PLCA cycles, including NodeID, NodeInfo, NodeSchedule, Hyper, and Standard Beacon PLCA cycles.
 - Assumption: Current PLCA devices will interop whether they receives 20,40, 60, 80, or 100-bits Beacon.
 - Need feedback on current PLCA devices in the market



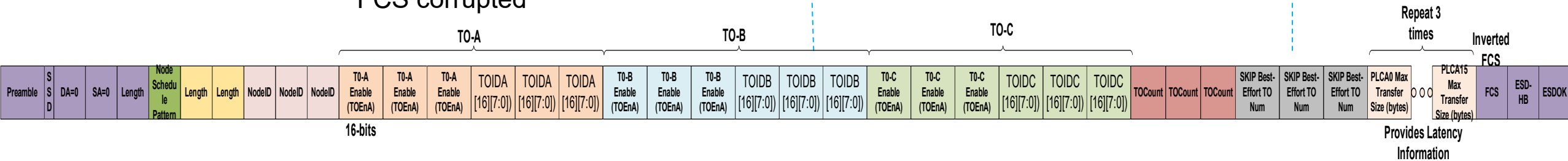
Plug and Play Support - Details



Standard Ethernet Packet Format with FCS corrupted



Standard Ethernet Packet Format with FCS corrupted



Informative

Note: Since 10baseT's error rate is 10^{-10} (one error every 1000 seconds), Redundancy (3) for the critical fields provides additional correction; 2 needs to match

Plug and Play Support – Dynamic LocalNodeID assignment

- Two Parameters: NumPLCANodes (to accommodate PLCA devices) and MaxID (could be fixed to 255)

ID_Is_Used[] = 0 // Array[MaxID-1:0]

For (i= 0 to MaxPLCANodes)

ID_Is_Used[i] = 1; // TO-0 + PLCA reservation

ID_Selected = 0;

Collision = 1;

While(Collision) // This is looped by Coordinator until no Collision seen {

Coordinator do NodeID-Beacon PLCA cycle and TOCount= MaxID {

Collision = 0;

If (ID_Selected == 0)

LocalNodeID = Pick an unused random ID between NumPLCANodes+1 and MaxID (ID_Is_Used[LocalNodeID] = 0) // Done by both Coordinator and Follower

If (TO != LocalNodeID) & COMMIT

ID_Is_Used[TO] = 1;

If (TO == LocalNodeID)

Send COMMIT

If collision

Collision = 1; Wait till end of PLCA cycle and pick an unused random ID between NumPLCANodes+1 and MaxID

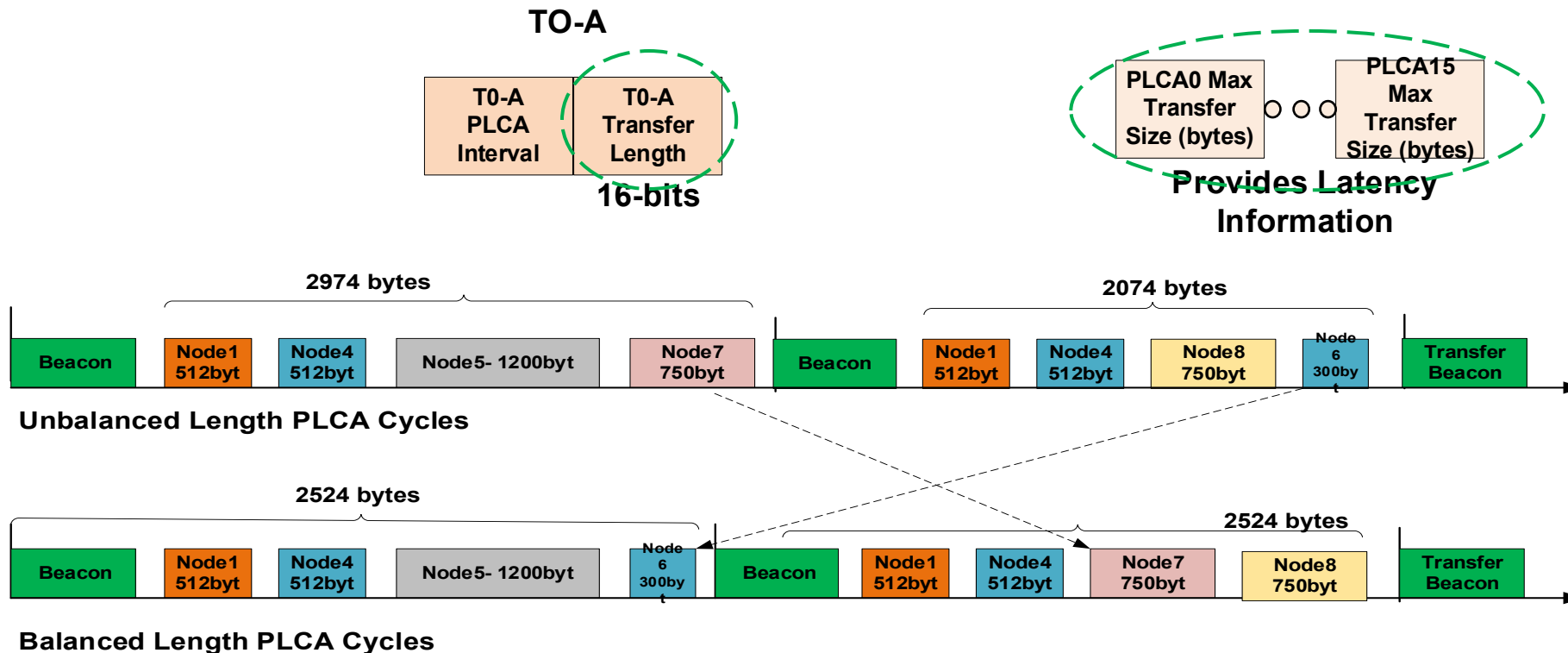
else

Keep LocalNodeID, ID_Selected = 1; }}

- Coordinator goes through NodeID-Beacon PLCA cycles till no Collisions seen and now it has list of all the NodeIDs used; and all the Nodes have unique LocalNodeIDs

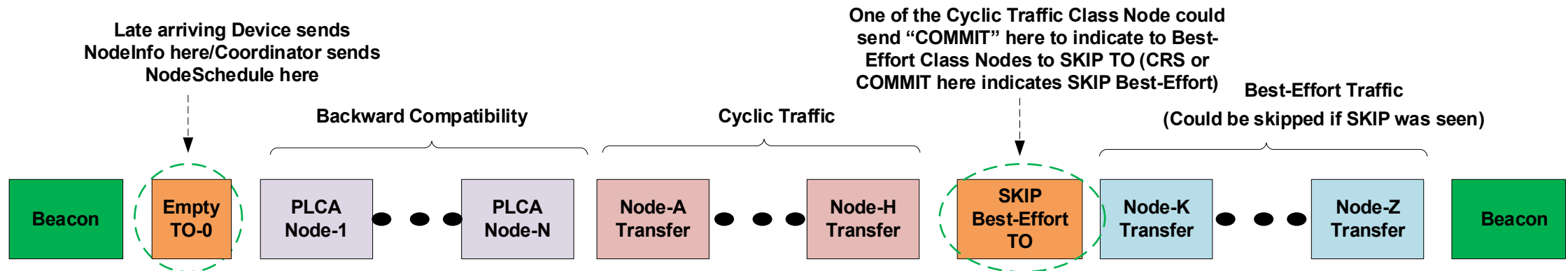
Plug and Play Support – Max Transfer Size Information

- Additional PLCA Optimization: If we continue to support large packets
 - In addition to A, B, C PLCA Intervals, a Node can also provide “Max Transfer Size” for each traffic class (includes header bytes).
 - The scheduling algorithm could use it to balance each PLCA cycle duration.
 - The Coordinator also sends the “PLCA Max Transfer Size” in the Schedule BeaconCycle; informative, so Nodes can calculate latency

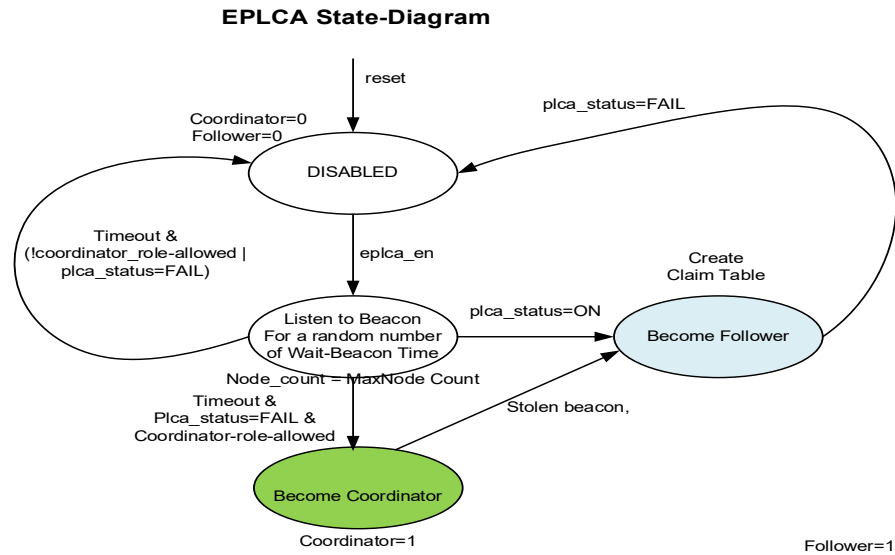


Plug and Play Support – Skipping Best-Effort Traffic and Nodes Joining/Powering-up Late

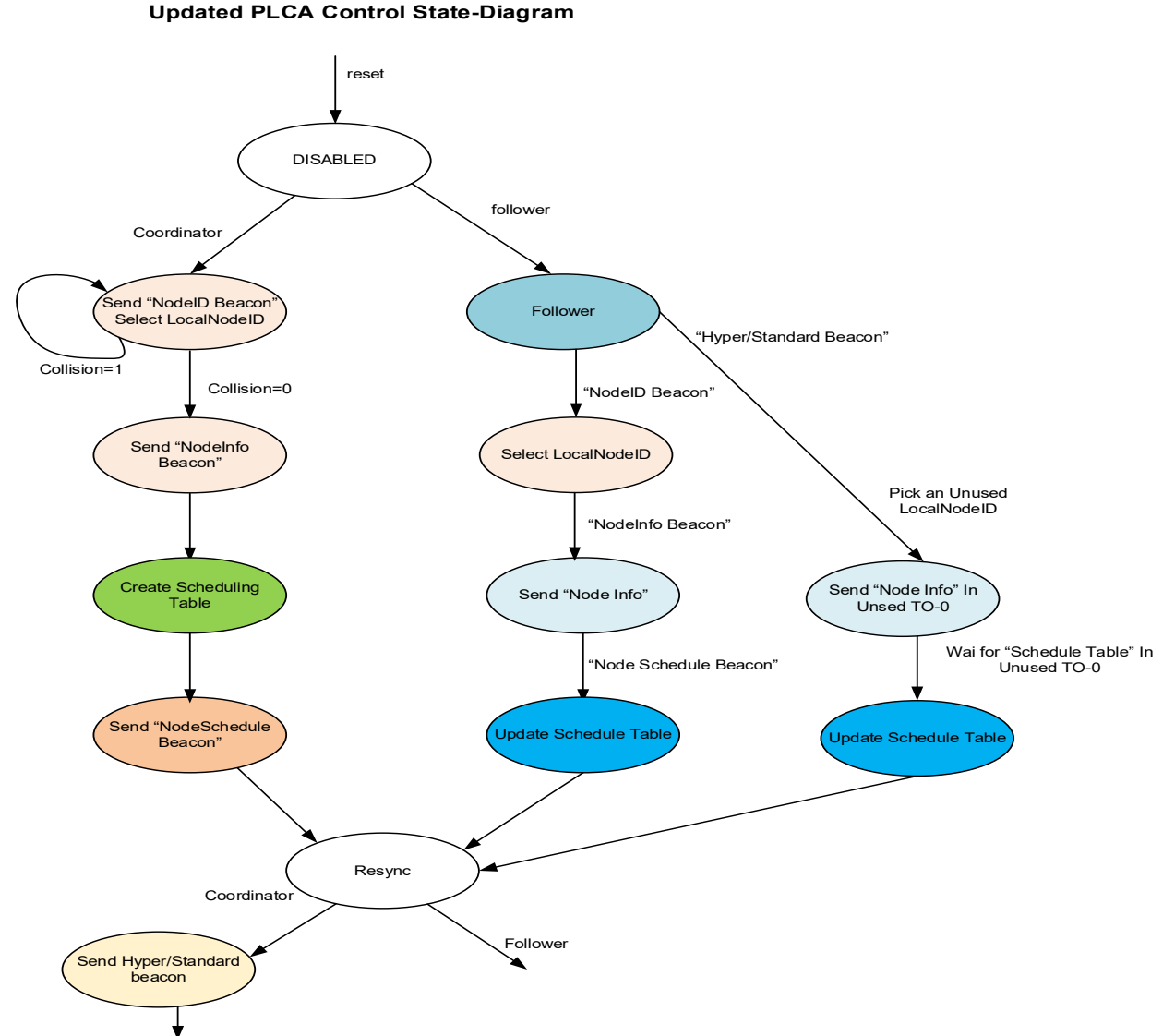
- The Best-Effort traffic in a PLCA cycle can be skipped if any Cyclic-Traffic Class Node sends COMMIT in “SKIP Best-Effort TO”
 - Coordinator will start beacon after “SKIP Best-Effort TO” and the Best-Effort Nodes will also skip the their TOs on seeing COMMIT or CRS in “SKIP Best-Effort TO”
- Handling Devices joining late or powering up late – New problem, not in PLCA mode
 - Self Powered devices will power up differently
 - Option-1: Coordinator waits XX ms before doing NodeID-Beacon
 - Option-2: A Node joining later can indicate that it is joining in unused TO-0?
 - Late joining PLCA device can join only as Best-Effort Node, since “SKIP Best-Effort TO number” is already communicated to others. Still Not a clean solution.



Plug and Play Support – High Level State-Diagram



Similar to current DPLCA StateDiagram



Rest of the Current PLCA flow

Diagram illustrating a message flow:

- A message labeled "Node Info In Unused TO-0" is sent to a green oval.
- The green oval contains the text: "Update Scheduling Table. Send 'Schedule Table' in Unused TO-0."

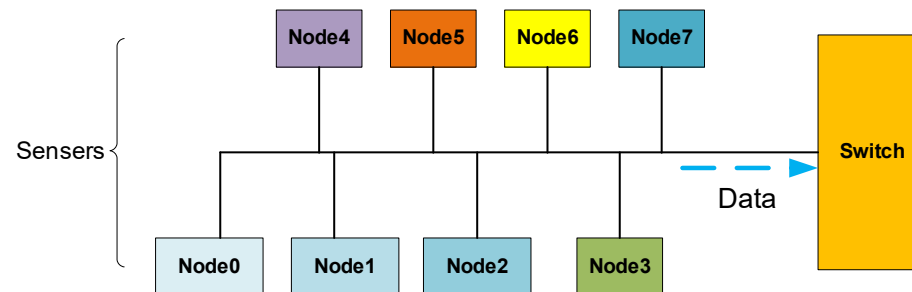
Plug and Play Support

- Pros

- Dynamic Node discovery and automated scheduling; if a Coordinator drops off, another node can take over and send beacons (no need to redo scheduling)
- Number of TO's in a PLCA cycles is smaller than total number of Nodes*Traffic-Classes
- Skip mechanism for On-Demand Traffic Class; allows more low-latency Cyclic-Traffic Nodes
- Compatible to PLCA (and CSMA/CD?)

- Cons

- Not compatible to D-PLCA (CoordinatorRole and NodeID assignments are automated/dynamic like D-PLCA)
- No Ageing support
 - Do we need this as we have already reduced the number of TOs? Aging will affect latency predictability and adds collisions
- 12.5 Mbps half-duplex medium is still going to be the limiting factor for low-latency, large-data, multi-node applications

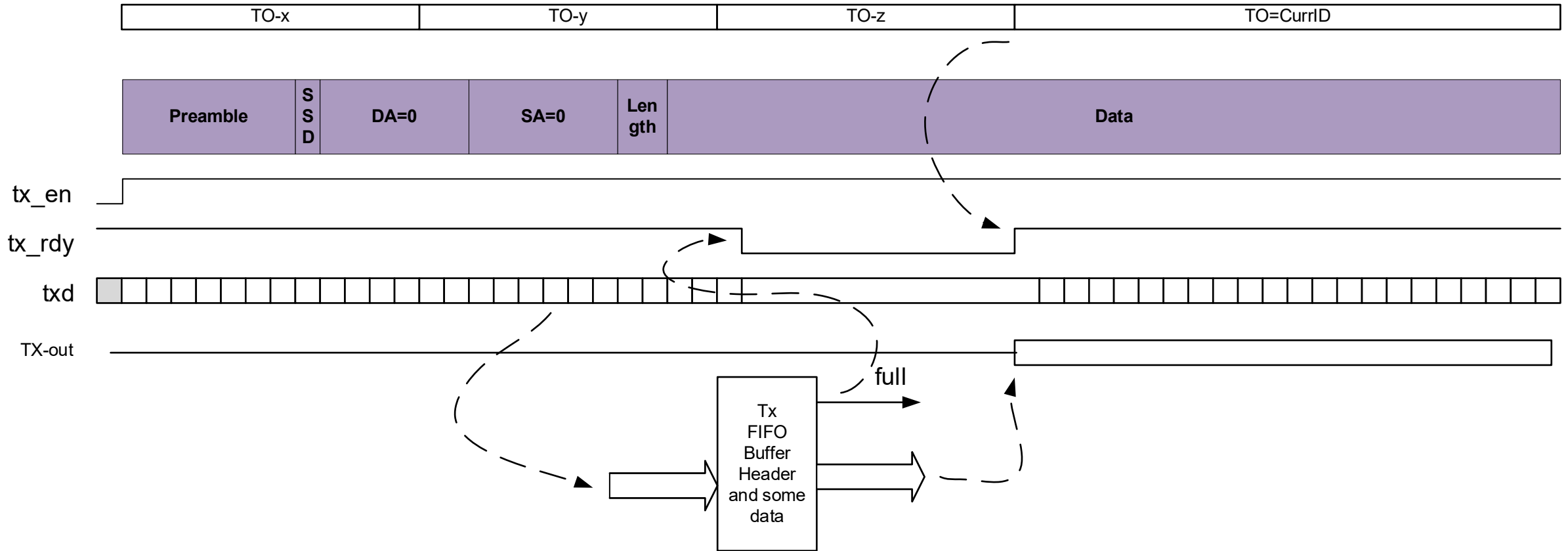


Sensors may not send data periodically but require low latency when on-demand data needs to be sent and hence may end up requiring TO allocation in every PLCA cycle

Bandwidth Based Latency Improvements – With additional changes outside PLCA

- 10BASET1's latency also depends on the bus bandwidth/speed. CAN XL already supports 20Mbps.
- Optimization-1: Instead 4b/5b encoding implement 128b/132b encoding - Make this optional feature (backward compatibility)
 - 25% overhead (vs) 3.1% overhead
- Optimization-2: Header Optimization?
 - The current Header overhead is 39 bytes; This is 18% ($39/217 * 100$) overhead for a 256-byte packet; 8.3% ($39/473 * 100$) overhead for a 512-byte packet
 - 8-byte Preamble & SFD + 6-byte Destination MAC + 6-byte Source MAC + 2-byte Type/Length + 4-byte Frame Check Sequence (FCS) + 1-byte (ESD & ESDOK) + 12-byte Inter Packet Gap (IPG)
 - For the automotive zonal architecture, this could be optimized – Make this optional feature (backward compatibility)
 - 3-byte Preamble & SFD (could be still 8-bytes at the MAC interface)
 - 1-byte Destination and 1-byte Source MAC address (1-byte-EPLCA address to 6-byte MAC address translation table reside in EPLCA device)
 - 0-byte Inter Packet Gap (IPG) (12-byte could be introduced at the receiver MAC interface, if needed)
 - Total saving = 27 bytes; ~13% improvement for 256-byte packet (~6% for 512-byte)
 - This is 4.9% ($12/244 * 100$) overhead for 256-byte packet; 2.4% ($12/500 * 100$) overhead for 512-byte packet
 - MII Interface backward compatibility is an issue
 - Run the MII Rx interface faster to compensate for additional 27-byte padding overhead (preamble, address, & IPG); 2.5X times faster to handle 1-byte packets
 - MII TX interface is still an issue. In addition to TX_EN and TX_ER, we need TX_RDY to buffer and freeze the TX interface and avoids fake manipulation of collision/csr indications and retries even though there is no collision.
 - This also improves bandwidth a bit more; currently we send COMMITs while waiting for data from MAC even though we already have the TO.
 - Improvement: Instead of 8 * 300-bytes (~2400-bytes), we can do 8 * 400-bytes (~3200-bytes) in a 2ms PLCA.

MII – TX Path Update



New tx_rdy signal input to MAC; better control mechanism than CSR/COL as there is no collision in PLCA

Backup



Reference Presentations

- Modulo Operation based Diff Serc- Markus Jochim, NXP Semiconductors
 - https://www.ieee802.org/3/EPLCA/public/260827/jochim_eplca_01_260827.pdf
- Toyota – PLCA vs CAN, latency and system comparison:
 - https://standards.ieee.org/wp-content/uploads/2025/10/D1_02_Yuki-Wakida-Feasibility-Study-of-Scalable-In-Vehicle-Network-Deployment-FINAL.pdf
- onsemi – 40 node demonstration
 - <https://www.eenewseurope.com/en/first-phymac-controller-for-10base-t1s-single-pair-ethernet/>
- IEEE – PLCA comparison with CAN, CAN-FD and FlexRay
 - <https://ieeexplore.ieee.org/document/9277440>

