

OpenLcbCLib

C Library to build OpenLcb/LCC Nodes on any hardware

Lesson 2 - Protocol Basics

Jim Kueneman 12/31/2025

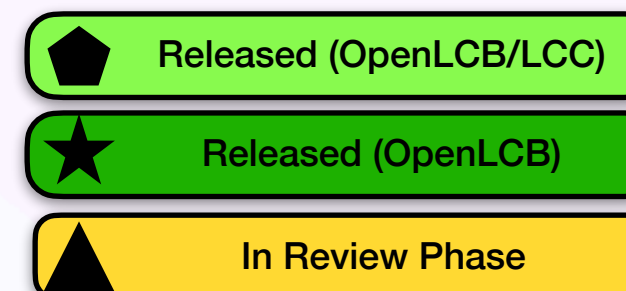
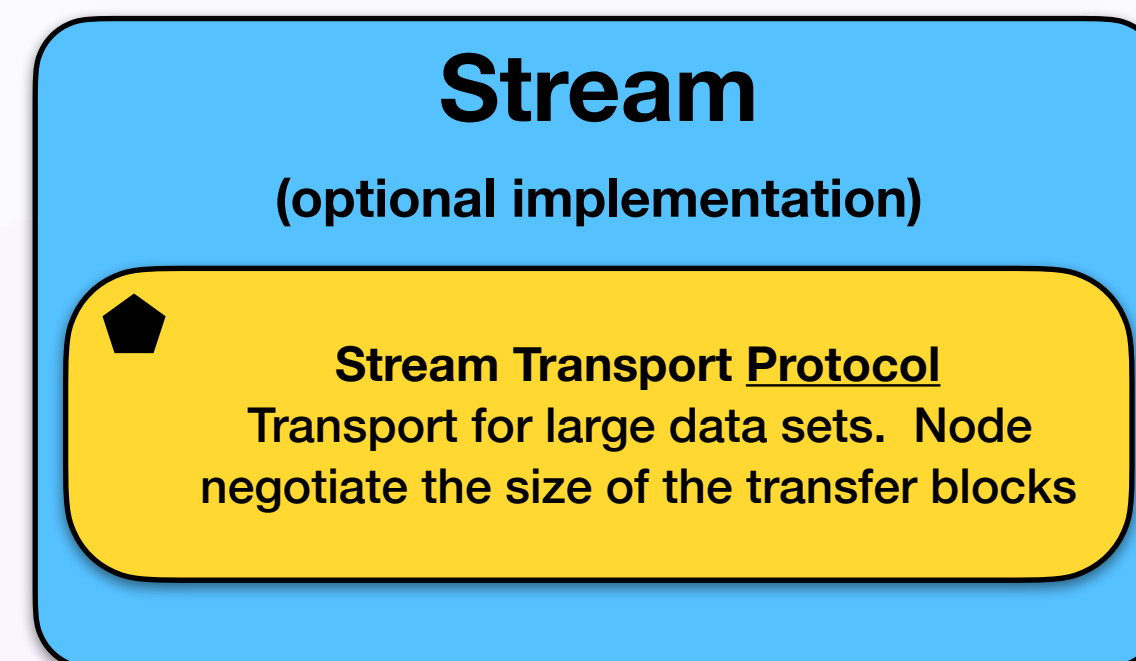
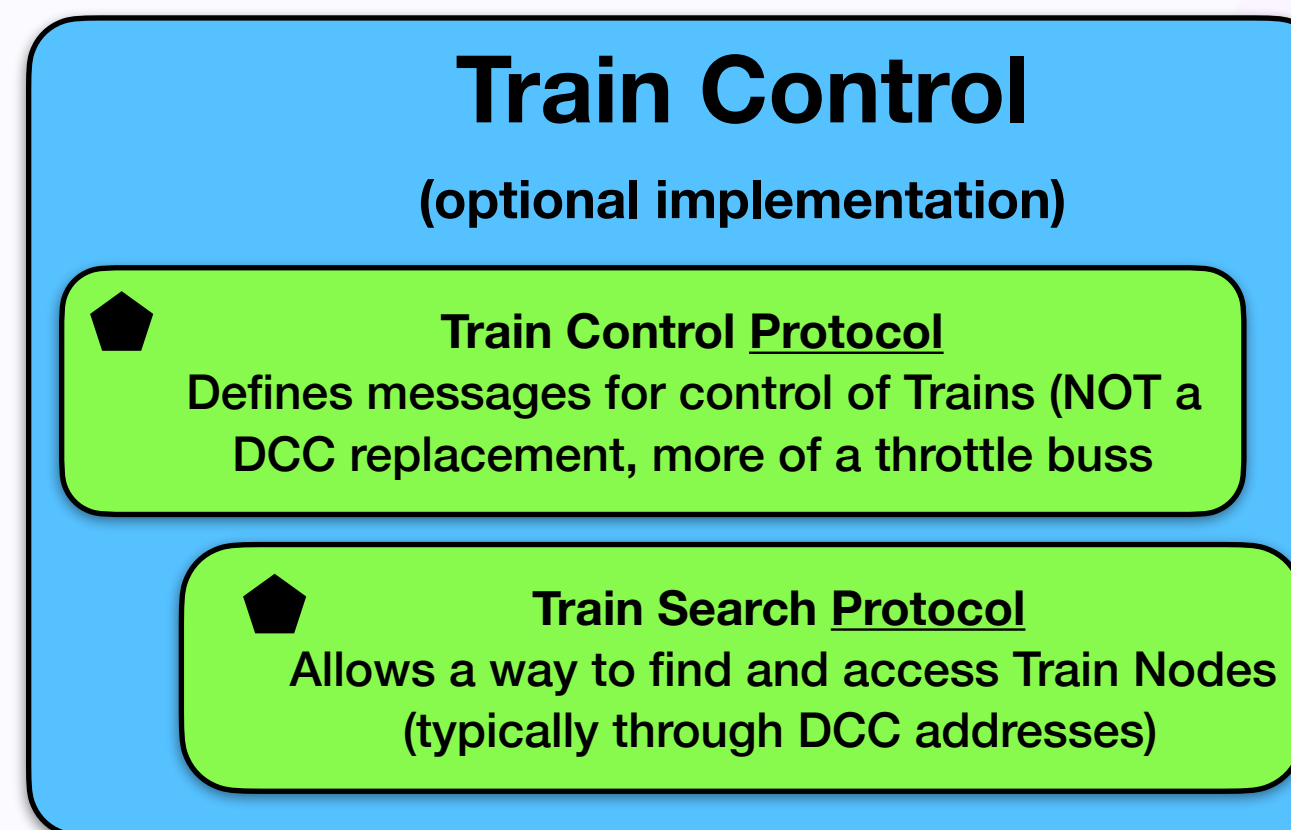
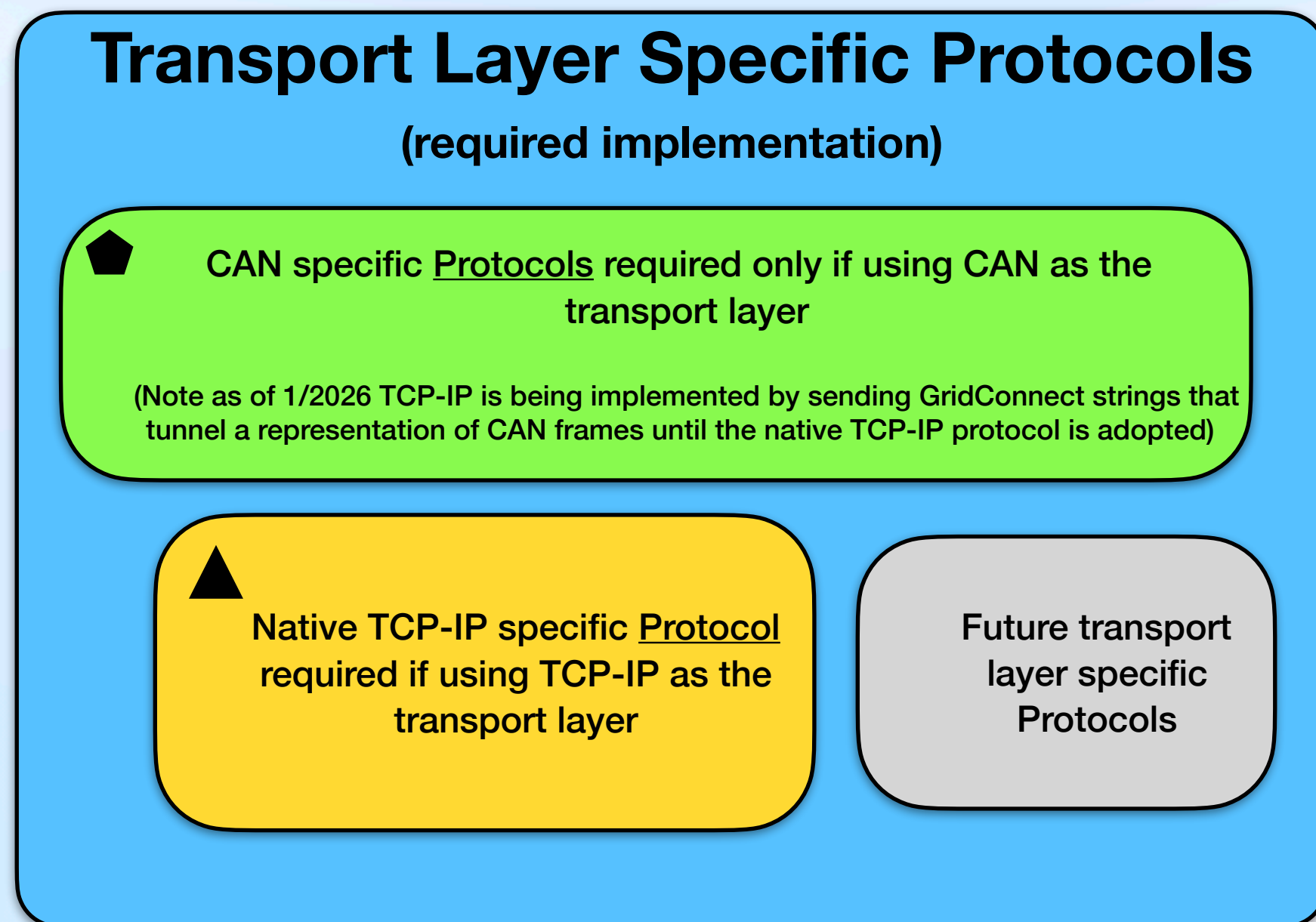
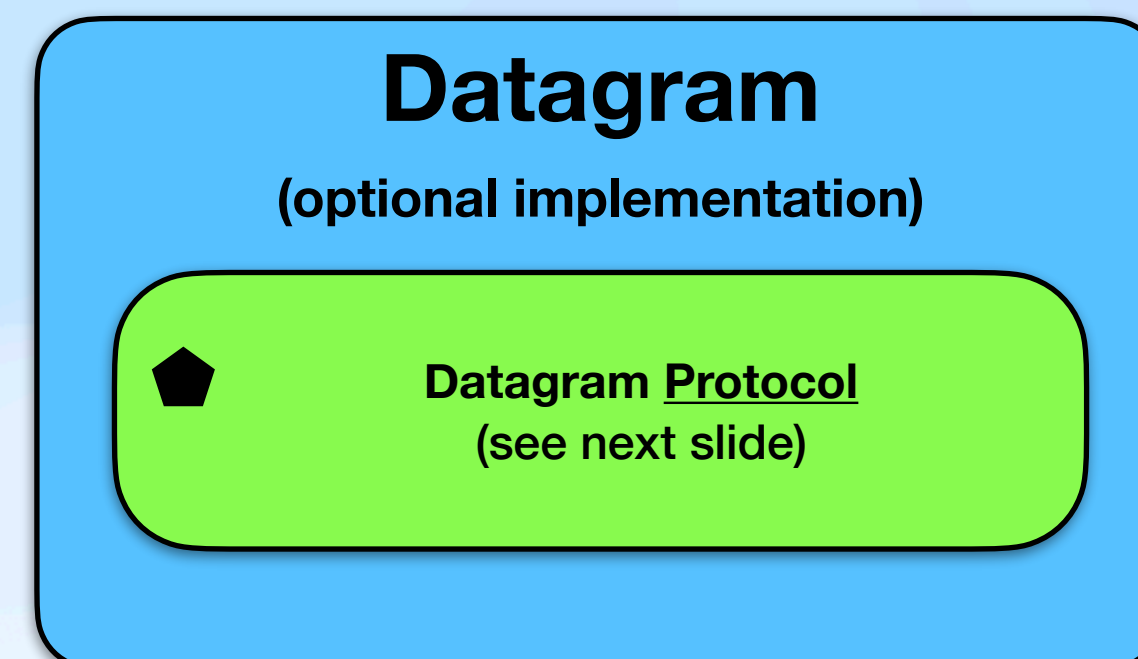
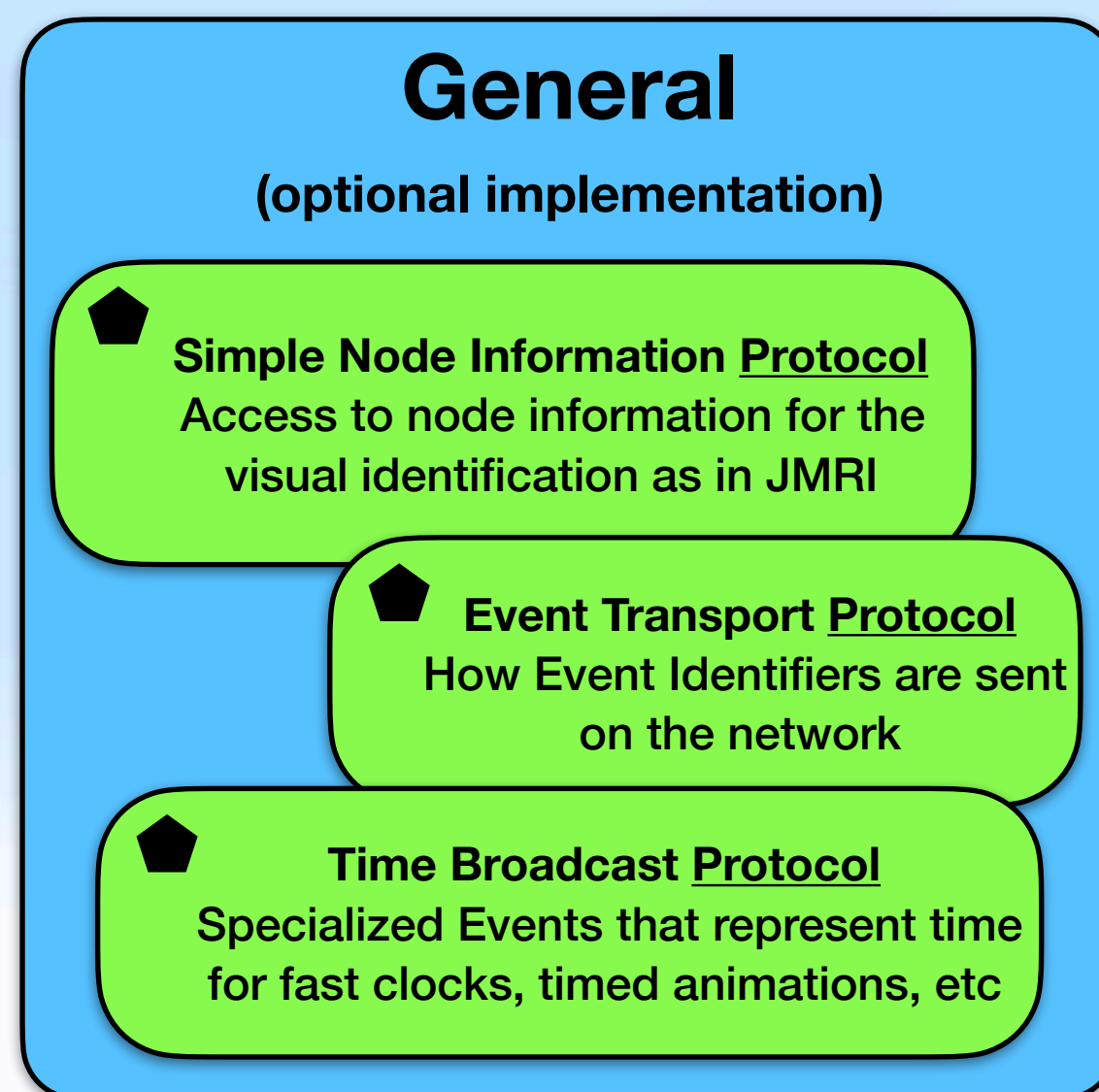
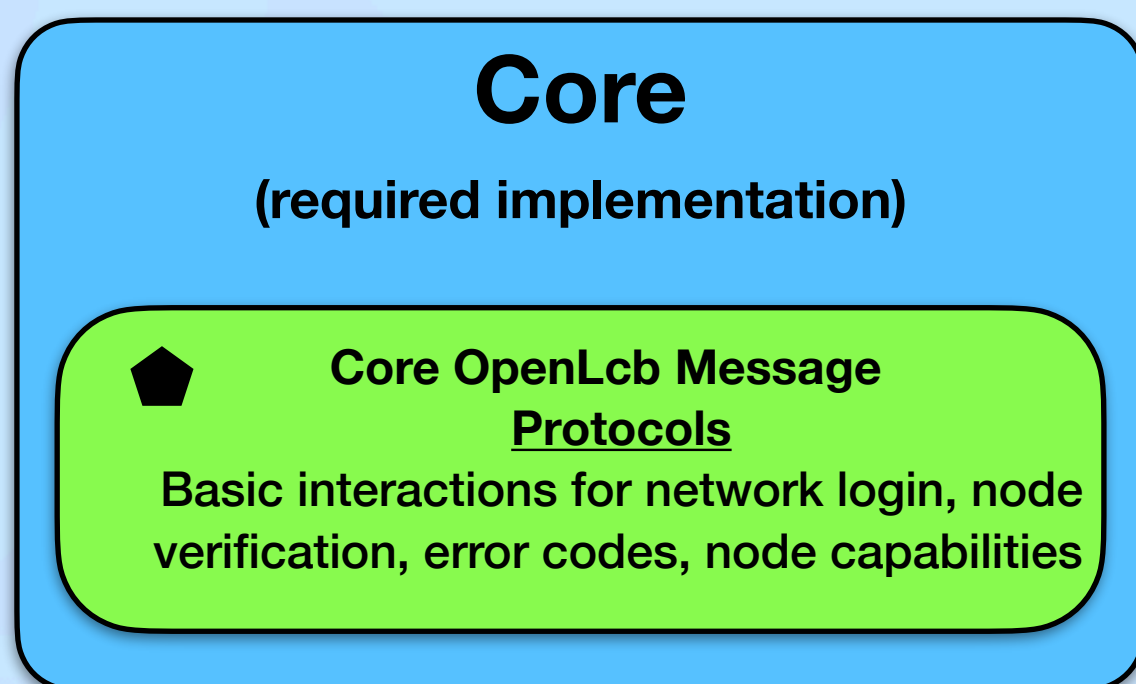
OpenLCB/LCC Protocols Define Node Communication

Applications require a high level understanding of the protocols to know what messages to handle and send

- NMRA Layout Command Control is a subset of OpenLCB
 - The adopted standards for LCC are here: <https://www.nmra.org/lcc>
 - The adopted and proposed standards for OpenLCB are here: <https://github.com/openlcb/documents>
- OpenLCB standards define the messages that are sent between nodes
- There are core OpenLCB message transfer mechanisms that are agnostic to what the physical transport layer is (CAN, TCP-IP, etc)
- There are adaptations to using CAN as the transfer mechanism
 - OpenLCB messages can require transferring large blocks of data and classic CAN data transfers are only 8 bytes; the OpenLCB message must be broken up in to smaller chunks and reassembled on the other end
 - Native TCP-IP transfer is in a draft state as of 1/2026 but will allow complete OpenLCB message to be sent in one blob

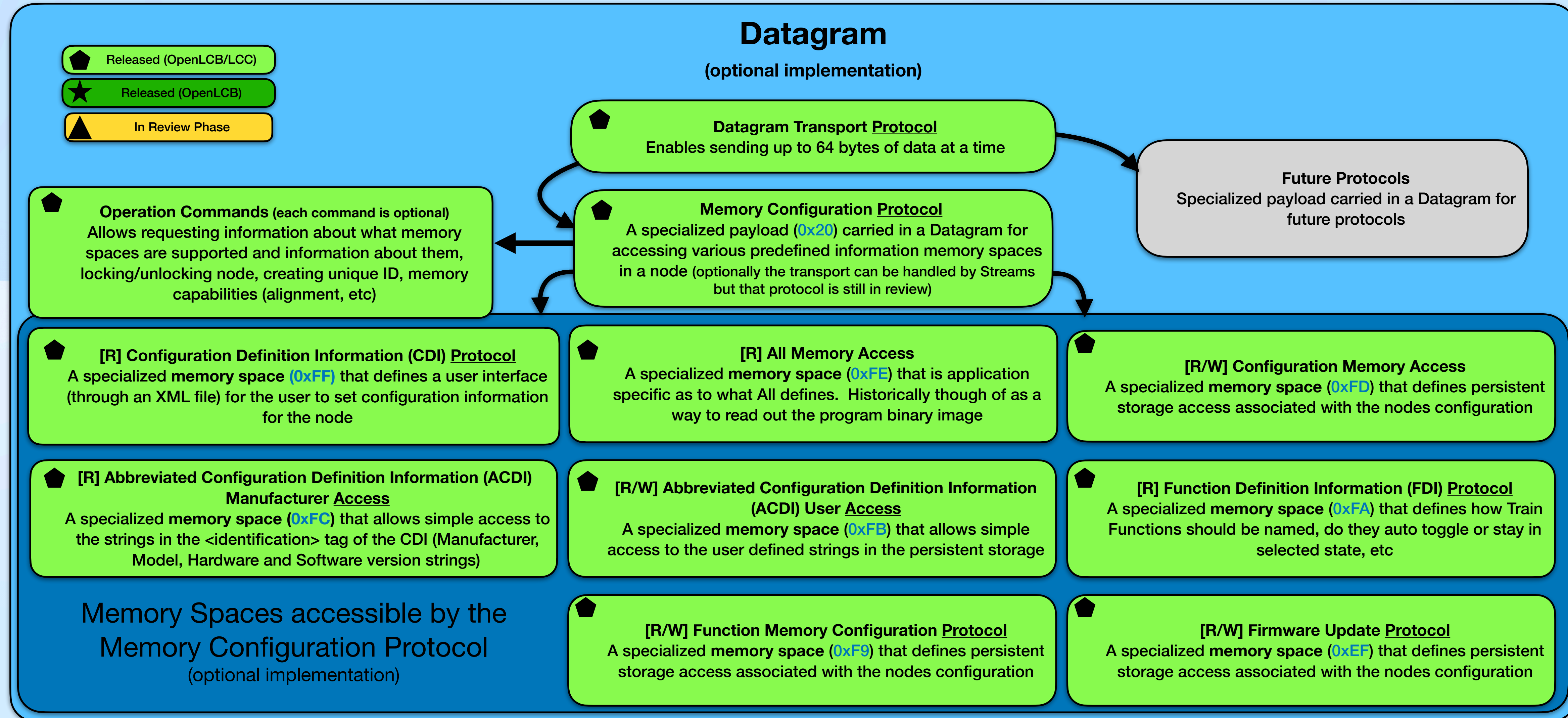
OpenLCB/LCC Protocol Overview

Applications define what optional protocols need to be implemented



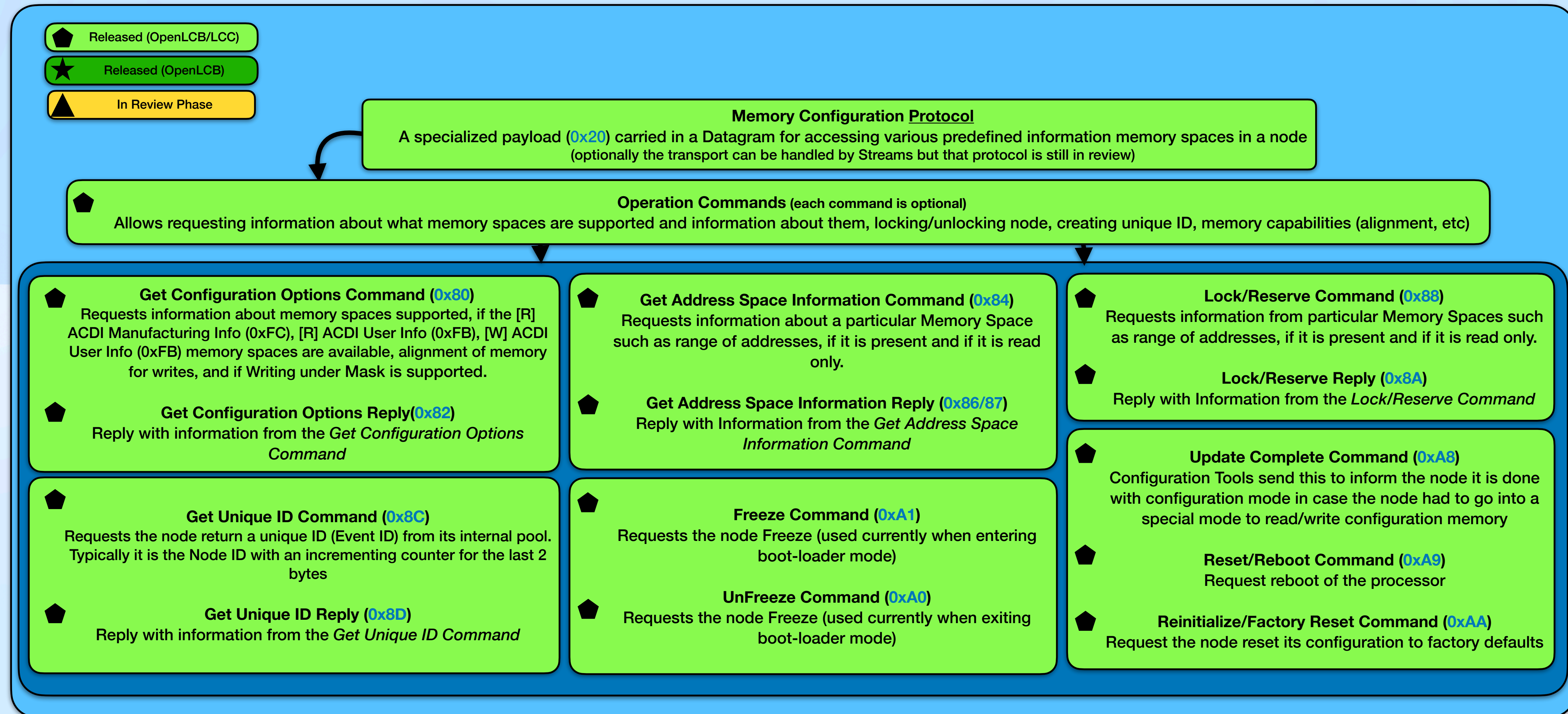
OpenLCB/LCC Datagram Protocols

Most Complex of All Protocols



Memory Configuration Operations Commands

Queries a node for capabilities and supported memory spaces



Designing a Node

Defining what a CAN based node will implement

Core

(required implementation)



Core OpenLcb Message Protocols

Basic interactions for network login, node verification, error codes, node capabilities

Transport Layer Specific Protocols

(required implementation)



CAN specific Protocols required only if using CAN as the transport layer

(Note as of 1/2026 TCP-IP is being implemented by sending GridConnect strings that tunnel a representation of CAN frames until the native TCP-IP protocol is adopted)

IN WORK

