OpenLCB Working Note	
Search	
Nov 30, 2020	**Preliminary**

1 Introduction

A Working Note is an intermediate step in the documentation process. It gathers together the content from various informal development documents, discussions, etc into a single place. One or more Working Notes form the basic for the next step, which is one or more Standard/TechNote pairs.

1.1 Served Use Cases

Each OpenLCB node can be uniquely identified and addressed by its 48-bit Unique Identifier. However, often the 48-bit identifier is unknown to the user, or at least inconvenient to determine. The search protocol will provide a means of node lookup by human readable string.

The results of the search would be a set of nodes that match the search criteria.

Though the idea would be to have a search protocol that can generically applied to different use cases, the use case that is front and center is the ability to find a train node for the purpose of bringing it under the control of a throttle.

In a traditional DCC based system, a train is identified by its [typically unique] 2 or 4 digit number. In North American prototypes, this can typically be the cab number on the locomotive. However, in Europe, a 4 digit number is not large enough to represent the 6 digit cab numbers used by the prototypes there. As a result, many DCC systems already on the market today in Europe allow the user to associate additional metadata with the DCC address (like name, prototype model, road number, owner, etc...) and search a centralized database to find the appropriate DCC address associated with a given model.

Given that OpenLCB extends the address space of train nodes much further beyond that of a DCC address, a search protocol is even more critical to the user being able to find the OpenLCB node representing a given model and bring it under control.

1.1.1 Train Control Use Cases

Find a node representing:

1. a DCC 2 or 4 digit addressed
2. a Marklin addressed
3. Other legacy protocol address (DC, DCS, TMCC, RailCommand, etc...)
4. User supplied metadata such as owner, road name, cab number, prototype model, etc...)

1.1.2 Design options

There are two design architectures currently under consideration. One of those architectures would require a centralized node that would index all the nodes that make up the search dataset and arbitrate responses based on a search query.

The second architecture option is to distribute the search to all the nodes that make up the dataset individually, and make each node participating in the dataset to independently formulate a response.

1.1.2.1 Centralized Index

The concept of requiring a specialized node that is not a traditional control endpoint is somewhat counter to the spirit of OpenLCB. Though not strictly prohibited, OpenLCB is biased toward distributed, peer-to-peer, interactions rather than centralized master/slave interactions.

The prime advantage of a centralized approach is that the responses can be throttled and provided in a predetermined order which is convenient for the user.

A centralized solution can exist without the exclusion of a distributed solution, but for the moment, this option will remain off the table.

1.1.2.2 Distributed Search

The distributed approach requires that each node that is contained in the dataset take an individual look at its attributes and respond if it finds that it matches the search criteria. The challenge with this approach is that the node performing the search will not have any control of the order in which the results are returned. The solution to this is that the requesting node will need to put the results in order as they come in.

1.2 Unserved Use Cases

2 Specified Sections

This is the usual section organization for a Technical Note, to accumulate the Standard and Technical Note content in its eventual order.

2.1 Introduction

Note that this section of the Standard is informative, not normative.

2.2 Intended Use

Note that this section of the Standard is informative, not normative.

2.3 Reference and Context

Lorem ipsum dolor sit amet, consectetur adipiscing elit.[1] Fusce ornare mattis justo vitae imperdiet. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

[1] See the “Common Information” OpenLCB Technical Note for detailed conventions on bit and byte numbering. Briefly, the least significant bit of a field is numbered with zero in OpenLCB descriptions, but note that other technologies may use other conventions.

2.4 Message Formats

In this section, we will look at some possibilities for the distributed search method.

2.4.1 Query

The search query should be a global message in order for each node[2]. Currently the only global message format defined is an event and contains only 8 bytes of payload. Here is one proposal on how it might be formatted as an event payload:

Byte 1	0x01 – Well-known
Byte 2	0x00 – Automatically routed
Byte 3	0x00: search all capable nodes, 0x01: search all train nodes, 0x02-0xFF reserved
Byte 4	Flags: Bits 7..6: 0x0 = exact match, 0x01 = <string>*, 0x10 = *<string>, 0x11 = *<string>* Bit 5: 1 = string response okay, 0 = datagram only Bits 4..0 = bit mask of metadata fields to search
Byte 5	1 to 4 character search string, null terminated if less than 4 bytes. Additional wild cards are denoted by the "*" character.
Byte 6	
Byte 7	
Byte 8	

The main drawback of using an event is that the search criteria is bound to a very small number of characters. The search string size is further limited if UTF-8 encoding is used.

An alternative would be to allow for a new kind of global message that can have more than 8 bytes of payload:

Byte 1	0x01 – Well-known
Byte 2	0x00 – Automatically routed
Byte 3	0x00: search all capable nodes, 0x01: search all train nodes, 0x02-0xFF reserved

[2] There should likely also be an addressed version of the query

Byte 4	Flags: Bits 7..6: 0x0 = exact match, 0x01 = <string>*, 0x10 = *<string>, 0x11 = *<string>* Bit 5: 1 = string response okay, 0 = datagram only Bits 4..0 = bit mask of metadata fields to search
Bytes 5 - 6	Max response size in bytes
Bytes 7 - 8	Search sequence number. Increment by 1 with each successive search. Used for tying results back to individual query.
Bytes 9 - 254	Null terminated UTF-8 search string. Additional wild cards are denoted by the “*” character.

Each node in the dataset receiving the global search message would perform an appropriate search of its own metadata fields and respond if it finds a match.

2.4.2 Response

Since the number of responses is theoretically unbounded, it is important that the the response traffic be relatively low priority. There are two possible forms the response could take:

1. Node replies with its 48-bit Unique Identifier requiring further inquiry from the requesting node for the contents of various searched fields.
2. Node replies with the contents of the searched fields as part of the reply

Some attention should be paid to message priorities. Generally speaking, replies should have a higher priority than requests. If traditional events are used for the search query, this potentially presents a problem as the replies would then need to be the highest possible priority. Since search information is naturally lower priority than other control traffic, this presents a problem.

2.4.2.1 Node ID Reply (option 1)

This has the advantage of simplicity for the replying node. Furthermore, if the requesting node intends to sort the responses solely on the basis of of its 48-bit node ID, further inquiry of the search fields only need be completed for those nodes that are in the current viewable focus of the user. Further inquiry on the details of the node can be completed with the memory configuration protocol.

Name	Dest ID	Simple Node	Common MTI	CAN format	Data Content
Node ID Search Reply	Y	Y	0xXXXX	0x19XX,Xsss	fddd, Sequence Number

One disadvantage is that if the requesting node wants to sort the responses by some criteria other than the Node ID, it must have at least enough storage to hold the Node ID of every single node that responds, at least until it can query the additional field(s) it wants to be the basis of its sorting. This is not necessarily a bound size as the dataset scales up to an indeterminate size.

2.4.2.2 Reply with Content (option 2)

This has the advantage of simplicity on the requesting node, without significantly increasing the complexity on the replying node. This could result in more flexibility on the field(s) used as the basis for sorting as well as the bind the memory required by the requesting node to handle the response.

Name	Dest ID	Simple Node	Common MTI	CAN format	Data Content
Node ID Search Reply	Y	Y	0xXXXX	0x19XX,Xsss	fddd, Sequence Number, list of null terminated metadata field contents up to <max response size> bytes.

One disadvantage is that more network bandwidth, perhaps significantly more, may be required for all the resulting nodes to respond. This may be acceptable as the network bandwidth should scale based on the number of nodes in it through additional segments, faster transport layers, etc...

2.5 States

2.6 Interactions

2.6.1 Case Study, Bridge to Legacy DCC style throttle

In this case, he legacy throttle request a 2/4 digit DCC address. The bridge makes an exact match search on the address. The DCC command station responds (as a proxy node) with the technology attribute set to “DCC”

> Note, other responses may also occur, but the bridge will ignore them unless the technology attribute is “DCC” and the address is an exact match.

Bridge makes a map between the DCC address and 48-bit Unique Node ID of the proxy node. The Bridge competes the assignment with the legacy throttle.

2.6.2 Case Study, Intermediate Throttle

This throttle has a finite number of resources, but is native OpenLCB. Example hardware specs might includes.

1. 32-bit MCU
2. less than 1MB FLASH
3. less than 256K RAM
4. 4 line LCD screen (graphic or character)
5. Numeric keypad + a few specially buttons

Throttle wants to make a query *123* on a large dataset. The result is 50+ nodes. The throttle keeps a scroll list that is 10 entries and can display 4 at one time on the display. As the results come in, they go into the sorted list of 10 entries. Those entries that get pushed out of the list are forgotten by the throttle, they are out of focus of the display and the scroll buffer.

Out of focus up scroll buffer, empty when focused on top of list
Out of focus up scroll buffer, empty when focused on top of list
Out of focus up scroll buffer, empty when focused on top of list
In display focus 1
In display focus 2
In display focus 3
In display focus 4
Out of focus down scroll buffer, empty when focused on bottom of list
Out of focus down scroll buffer, empty when focused on bottom of list
Out of focus down scroll buffer, empty when focused on bottom of list

When the throttle scrolls up in the list, it can immediately display the out of focus entries it has buffered. If this results in empty slots in the list, another search is kicked off in order to fill in the gap(s).

3 Background Information

Quisque sollicitudin tempor bibendum. Donec consectetur condimentum sollicitudin. Sed dignissim velit id felis lacinia at eleifend nisi laoreet. Vivamus tristique porta ornare. Vivamus feugiat dolor id lectus aliquet luctus.

Table of Contents

Title..1
Section Title..2
Section Title..2