OCORA

Open CCS On-board Reference Architecture

CCS Communication Network - Evaluation

Document ID: OCORA-TWS02-010

Version: 1.00

Release: Delta

Date: 30.06.2021

Management Summary

Today the interfaces between CCS components on the vehicle are proprietary. The proprietary interfaces do not allow to exchange or update CCS components from different suppliers. The OCORA architecture [7] aims for plug and play interchangeability within the CCS domain through the specification of a generic and open communication backbone, the CCS Communication Network (CCN) former called Universal Vital Control and Command Bus (UVCCB). The CCN itself will be modifiable in accordance with future technological evolutions by means of strict separation of the different communication layers (OSI Layers).

This document is based on the CCN evaluation report of gamma release [10]. It provides further investigations on data serialization formats, network architecture and cybersecurity.

The CCN evaluation report of gamma release [10] proposes the CCN to be a TSN Ethernet based network with the use of SDTv2 / SDTv4 as safety layer. In order to be able to integrate the CCN on the next generation of train communication network (NG-TCN) every hard-real-time CCS device (e.g. CCU, BTM etc.) should have at least one TSN-capable Ethernet port whereas for soft- or non-real-time CCS devices a single standard non-TSN-capable Ethernet port is sufficient. Hard-real-time devices can use both planes of NG-TCN with two TSN-capable Ethernet ports in order to improve reliability and availability.

On session layer TRDP 2.0, OPC-UA Pub/Sub (over TSN) or DDS/RTPS (over TSN) are suitable solutions. These three options will be further investigated considering the system architecture with platform/CCU and the subcomponents.

The proposed protocol stack of CCN is listed in the following table. Highly recommended standards to be used as reference for procurement in OCORA are listed in **bold** font.

<table>
<tr><th>Layer</th><th colspan="2">Protocol for hard-real-time data</th><th>Protocol for soft- or non-real-time data</th></tr>
<tr><td>(Safety Layer[1])</td><td colspan="3">(SDTv2 / SDTv4)</td></tr>
<tr><td>Session Layer</td><td colspan="3">TRDP 2.0, OPC-UA Pub/Sub or DDS/RTPS</td></tr>
<tr><td>Transport Layer</td><td rowspan="2"></td><td>UDP
TCP</td><td>UDP
TCP</td></tr>
<tr><td>Network Layer</td><td>IPv4</td><td>IPv4</td></tr>
<tr><td>Data Link Layer</td><td colspan="2">Time-Sensitive Networking (TSN)
IEEE 802.1</td><td>Standard Ethernet IEEE 802.3</td></tr>
<tr><td>Physical Layer</td><td colspan="3">100BASE-TX or 1000BASE-T</td></tr>
</table>

Table 1: Protocol Stack CCN

The defined protocol stack allows safety-related and hard-real-time data traffic. For non-safety-related and soft- or non-real-time applications, standard TCP/IP or UDP/IP data traffic over standard Ethernet (IEEE 802.3) can be used on the same physical layer of the CCN.

As a result of the evaluation of data serialization formats it is proposed to use a mix of Bitstream and JSON / XML over the CCN. For time-critical CCS-applications the interfaces will anyway be specified which allows Bitstreams as a non-self-describing but rather fast format. For non-time-critical applications such as maintenance, a human readable data format like JSON or XML would be well suited.

The investigation of network architectures on cybersecurity shows that the zoning concept of the onboard networks is not clearly defined yet. The zoning concept must be discussed and investigated in the subsequent phases of the OCORA initiative. Furthermore, the CCS and TCMS domains must elaborate the same understanding of the network architecture. At the end, the results shall be incorporated into the next version of IEC 61375 standard.

[1] Safety Layer is only applicable for safety-related data traffic.

Revision history

Version	Change Description	Initial	Date of change
1.00	Official version for OCORA Delta Release	SSt	30.06.2021

Table of contents

Table of figures

Table of tables

References

Reader's note: please be aware that the numbers in square brackets, e.g. [1], as per the list of referenced documents below, is used throughout this document to indicate the references to external documents. Wherever a reference to a TSI-CCS SUBSET is used, the SUBSET is referenced directly (e.g. SUBSET-026). OCORA always reference to the latest available official version of the SUBSET, unless indicated differently.

[1] OCORA-BWS01-010 – Release Notes

[2] OCORA-BWS01-020 – Glossary

[3] OCORA-BWS01-030 – Question and Answers

[4] OCORA-BWS01-040 – Feedback Form

[5] OCORA-BWS03-010 – Introduction to OCORA

[6] OCORA-BWS04-011 – Problem Statements

[7] OCORA-TWS01-030 – System Architecture

[8] OCORA-TWS02-020 – CCS Communication Network – Proof of Concept (PoC)

[9] OCORA-40-003-Beta – UVCC-Bus Evaluation, Version 1.01

[10] OCORA-40-003-Gamma – UVCC-Bus Evaluation, Version 2.00

[11] IEC 61375-2-3: Railway Applications – Electronic railway equipment – Train communication network (TCN) – Part 2-3: TCN communication profile, 2015

[12] IEC 62443-3-3: Industrial communication networks – Network and system security – Part 3-3: System security requirements and security levels, 2013

[13] CENELEC prTS 50701: Railway applications - Cybersecurity, Version D8E5

[14] CTA-T3.5-D-BTD-002-12_-_Drive-by-Data_Architecture_Specification

[15] OPC 10000-6 Part 6: Mappings, 2017

[16] RFC 8949, Concise Binary Object Representation (CBOR), 2020

[17] UIC 559, Specification "Diagnostic Data Transmission" from railway vehicles, 2010

1 Introduction

1.1 Purpose of the document

This document summarizes the work done in the workstream CCS Communication Network (CCN). It documents the evaluations on different communication levels and defines the CCN network embedded in the train.

This document is addressed to experts in the CCS domain and to any other person, interested in the OCORA concepts for on-board CCS. The reader is invited to provide feedback to the OCORA collaboration and can, therefore, engage in shaping OCORA. Feedback to this document and to any other OCORA documentation can be given by using the feedback form [4].

If you are a railway undertaking, you may find useful information to compile tenders for OCORA compliant CCS building blocks, for tendering complete on-board CCS system, or also for on-board CCS replacements for functional upgrades or for life-cycle reasons.

If you are an organization interested in developing on-board CCS building blocks according to the OCORA standard, information provided in this document can be used as input for your development.

1.2 Applicability of the document

The document is currently considered informative but may become a standard at a later stage for OCORA compliant on-board CCS solutions. Subsequent releases of this document will be developed based on a modular and iterative approach, evolving within the progress of the OCORA collaboration.

1.3 Context of the document

This document is published as part of the OCORA Delta release, together with the documents listed in the release notes [1]. Before reading this document, it is recommended to read the Release Notes [1]. If you are interested in the context and the motivation that drives OCORA we recommend to read the Introduction to OCORA [5], and the Problem Statements [6]. The reader should also be aware of the Glossary [2] and the Question and Answers [3].

1.4 Renaming

The CCS Communication Network was formerly called Universal Vital Control and Command Bus (UVCCB). The evaluations on different communication layers during beta [9] and gamma phases [10] concluded to use a time-sensitive Ethernet network as communication backbone. Therefore, the UVCC-Bus was renamed to CCS Communication Network.

1.5 Problem Description

Today the interfaces between CCS components on the vehicle are proprietary. The proprietary interfaces do not allow to exchange or update CCS components from different suppliers. The vendor locking created by proprietary interfaces leads to high costs. The existing proprietary interfaces do not allow to add new functions.

Moreover, these interfaces are implemented using heterogeneous bus technologies. This leads to increased complexity and extensive effort for the operator/maintainer to handle these heterogeneous systems.

1.6 Concept

The OCORA architecture [7] aims for plug and play interchangeability within the CCS domain through isolation of specific functions in combination with the specification of a generic and open communication backbone, the CCS Communication Network (CCN). In the following figure the final physical view of the OCORA architecture [7] is shown. The CCN connects all components of the future CCS system. The most important CCS components are:

- CCS computing units (CCUs)
- Driver Machine Interfaces (DMIs)
- Vehicle Locator (VL)
- Balise Transmission Module (BTM)
- Loop Transmission Module (LTM)
- National Train Control System (NTC) or Specific Transmission Module (STM)
- Cab Voice Device (CVD)
- Train Recording Unit (TRU)
- Input / Output Ports (I/O Ports)
- Gateway to Train Control Management System Network, Operator Network or Communication Network (ECN/ECN Security Gateway)

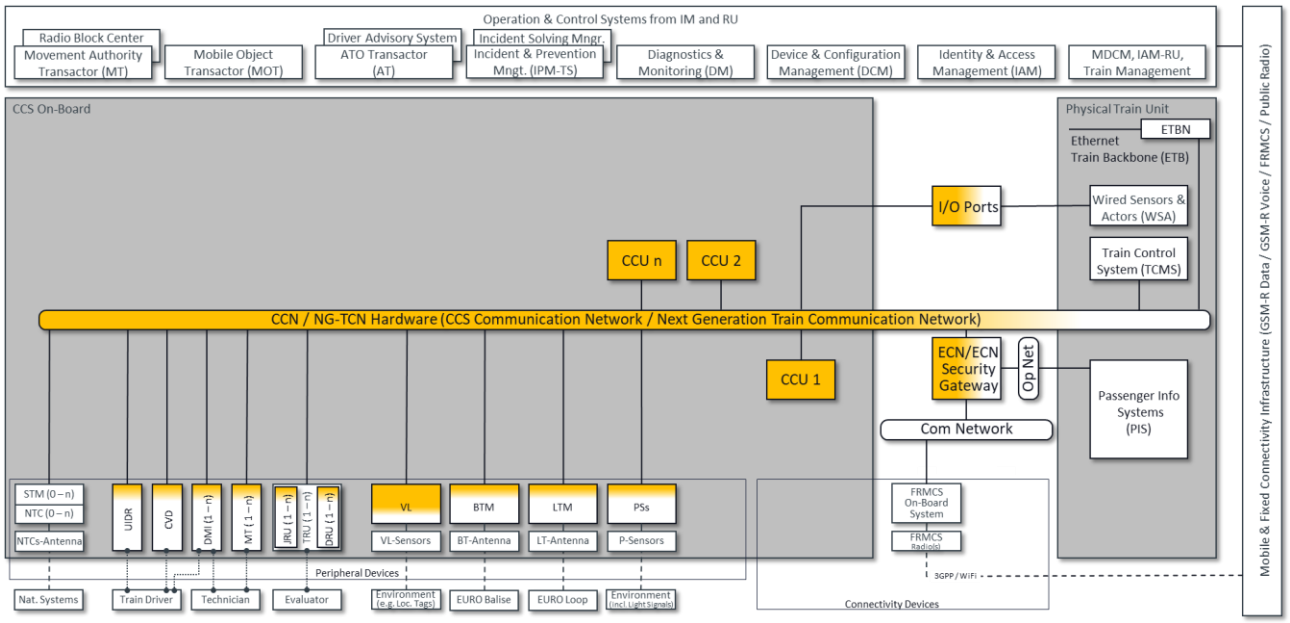

Figure 1: Technical architecture from [7]

In the final vision of the system an open standardized CCN (OSI-Layers 1 to 7 & Safety Layer) ensures the safe data connection between all CCS components. The bus topology allows simple upgrades of the CCS System by new functions or components. It also enables procurement on a component-based way which leads to more flexibility in the life cycle management and optimal components due to larger market size. The CCN itself will be modifiable in accordance with future technological evolutions by means of strict separation of the different communication layers (OSI Layers).

The CCN evaluation report of gamma release [10] proposes the CCN to be a TSN Ethernet based network with the use of SDTv2 / SDTv4 as safety layer. In order to be able to integrate the CCN on the next generation of train communication network (NG-TCN) every hard-real-time CCS device (e.g. CCU, BTM etc.) should have at least one TSN-capable Ethernet port whereas for soft- or non-real-time CCS devices a single standard non-TSN-capable Ethernet port is sufficient. Hard-real-time devices can use both planes of NG-TCN with two TSN-capable Ethernet ports in order to improve reliability and availability.

On session layer TRDP 2.0, OPC-UA Pub/Sub (over TSN) or DDS/RTPS (over TSN) are suitable solutions. These three options will be further investigated considering the system architecture with platform/CCU and the subcomponents.

The proposed protocol stack of CCN is listed in the following table. Highly recommended standards to be used as reference for procurement in OCORA are listed in **bold** font.

Layer	Protocol for hard-real-time data		Protocol for soft- or non-real-time data
(Safety Layer[2])	**(SDTv2 / SDTv4)**		
Session Layer	TRDP 2.0, OPC-UA Pub/Sub or DDS/RTPS		
Transport Layer		UDP TCP	UDP TCP
Network Layer		IPv4	IPv4
Data Link Layer	**Time-Sensitive Networking (TSN) IEEE 802.1**		**Standard Ethernet IEEE 802.3**
Physical Layer	**100BASE-TX or 1000BASE-T**		

Table 2: Protocol Stack CCN

The defined protocol stack allows safety-related and hard-real-time data traffic. For non-safety-related and soft- or non-real-time applications, standard TCP/IP or UDP/IP data traffic over standard Ethernet (IEEE 802.3) can be used on the same physical layer of the CCN.

1.7 Goal

The first aim of the delta phase of the workstream TWS02 CCS Communication Network is to investigate network architecture with detailed technical implementation of CCN in NG-TCN (network configuration), cybersecurity, new standards and regulations as well as an evaluation of data serialization formats.

Further, a demonstrator shall show the feasibility of the CCN as a logically separated network on a common physical train communication network (NG-TCN). The composition of the demonstrator helps to investigate the technical implementation details of the CCN in NG-TCN.

The following tasks have been performed during delta phase:

- **Evaluation CCN**
    - **Network Architecture**: Network architecture regarding connections from train to trackside (mobile communication gateway) must be further investigated together with TCMS domain / CONNECTA. Also, DMI concept with different displays from different domains on the driver desk must be clarified in order to define network architecture with its zones and conduits
    - **Detailed technical Implementation of CCN in NG-TCN**: The detailed technical implementation of CCN in NG-TCN shall be elaborated together with CONNECTA. The composition of the demonstrator (WP2) helps to investigate the technical implementation details.
    - **Cybersecurity**: The impact of the security concept of CONNECTA (NG-TCN) on the CCN shall be investigated. As a result of the investigation, the security requirements on the CCN (layers 1 to 6) shall be defined.
    - **New Standards and Regulations (e.g. TSI 2022)**: The work done in different working groups (e.g. IEC TC9 WG43 or ERA TWG Archi) shall be aligned in order to get consistent new standards and regulations (e.g. IEC 61375, TSI 2022, ERA Subsets, OCORA specifications)
    - **Evaluation of Data Serialization Formats**: Data serialization formats of application data shall be evaluated. Possible solutions are: bitstream like in today's subset specifications, XML, JSON, CBOR, Apache Thrift, Protobuf.

[2] Safety Layer is only applicable for safety-related data traffic.

- **Proof of Concept / Demonstrator CCN**: A demonstrator shall show the feasibility of the CCN as a logically separated network on a common physical train communication network (NG-TCN). Also, the impact of TSN-Ethernet versus non-TSN-Ethernet in congested network situation shall be demonstrated. The composition of the demonstrator helps to investigate the technical implementation details of the CCN in NG-TCN.

This document contains the current results of the evaluation tasks. The Proof of Concept / Demonstrator CCN part is documented in [8].

2 Serialization formats

2.1 Introduction

At the interface between the applications and the communication network lies the problem of serializing the data. The way data is handled and structured by applications is application specific and depends on the choice of programming language and implementation. For different applications to be able to communicate data in a generalized manner, as well as transforming the data objects into a format that can be sent over a serial network, the data is transformed into a cross platform format. This process is called serialization as it also permits the data to be sent over a serial communication interface without losing information or creating ambiguities. The words serialization and encoding will be used interchangeably in the following chapters.

The question of serialization needs to be addressed and specified at the application level. However, in this document a recommendation for a data serialization format from the point of view of the lower networking layers (1-6 in the OSI-Model) is provided alongside an overview of different serialization formats. The aim is to assure the compatibility between the applications using the CCN and the CCN.

The serialization formats differ from each other according to following criteria which will be used to evaluate them:

- Readability by humans
- Data typing
- Performance of encoder/decoder
- Space needs of serialized data (directly linked to networking speed)
- Platform / language independence
- Availability of implementations
- Open / proprietary
- Flexibility for upgrades
- Complexity of supported data structures

Some of these criteria should be given more importance than others, dictated by the needs of the CCN. These are briefly discussed below.

It is important to note that the data sent over the CCN will not have a complex structure. The data structures to be sent over the CCN are mostly a small number of variables that do not include more complex structures such as arrays of varying length. E.g. SUBSET-119 uses only following variable types:

- BOOLEAN1
- UNSIGNED8
- INTEGER16, 2s complement
- BITSET8
- UNSIGNED16

For clarity or more information, one can imagine grouping together certain of these variables to give the data some structure. For instance, all variables pertaining to track conditions could be grouped in a structure and separated from other variables. Moreover, the telegram can thus be structured by separating the fields data from a telegram header and safety trail.

The CCN will be a TSN Ethernet based network ([10]) and uses thus Ethernet frames. The payload data per frame is thus limited to the 1500 octets of a standard Ethernet frame. Looking at IEC 61375-2-3 ([11]) and SUBSET-119, along with the data one would send a header with information about the function the data is sent to, the function that created the data, a version information and message type (process data, message data) and a safety trail. The additional information needs to be included in the data frame and some of the 1500 available octets could be taken up by other protocols.

The system will not need remote procedure call functionality as the nodes will only communicate via a specified

network interface.

The CCN is designed to be the communication bus of choice for CCS Systems for the foreseeable future. The choice of serialization format should thus be robust over the next 30-40 years and be flexible enough to allow for adaptations to the data and protocols that could arise during this time.

2.2 Data formats

2.2.1 Bitstream

This method of serializing data is currently used for data transmission over the CCS bus. The data is serialized and transmitted using a bitstream. The format of the data is chosen when specifying the network. This specification does not need to follow a set of rules, as long as it is not ambiguous. It is however essential that all components adhere to the specification for the system to work smoothly. Using custom encodings and the knowledge that every node adheres to the specification (e.g. Subset 119), the data can be represented in a very compact way.

However, as the encoding is not standardized, a custom encoder and decoder needs to be developed for each system that communicates on the network as well as for each development environment used by the applications. This is only feasible for small systems involving a small number of nodes and applications that communicate a limited amount of data. Once set up, the format is also quite static, as changes to the specification can affect all the components. This can be slightly improved by including reserve bits set aside for future use. However, the data structure will remain very rigid. The advantage of this rigidity however is that the communication over the bus is well-defined and the encoding and decoding can be tailored to the needs of the application and thus be more performant. It is also easy to recognise and discard a message that does not follow any specified data structure.

2.2.2 XML

Extensible markup language (XML) is widely used for the representation of arbitrary data structures. It is simple, human readable and very general. Specified by the World Wide Web Consortium (W3C) in free open W3C recommendation, the language is very accessible and used in a wide range of applications. (http://www.w3.org/TR/xml/, version 1.0, last issue at time of writing: 2008)

XML is a textual data format i.e., the data structure as well as the data are represented as text. Encoding of the text using Unicode standards is supported. Typically, this is UTF-8 or UTF-16. Other encodings (ASCII, …) can be used but are not necessarily supported by every XML parser.

Thanks to its wide use, encoders and decoders are available as APIs for most of the programming languages.

Document type declaration (DTD) (with element type declarations) and schemas can be used to restrict the data types and format.

Due to the structure of XML (use of tags to delimit data) and the use of text (1 byte per character in UTF-8 for most common (i.e., ASCII) characters) this format is easily readable by humans, can however be very verbose and use a lot of space. Though it was designed to be used over the internet, it is not the most efficient format to send data over a network.

The UIC 559 Specification "Diagnostic Data Transmission" from railway vehicles [17], specifies the use of XML with an XML schema definition for diagnostic data transmission from railway vehicles to ground IT systems.

Typical syntax with DTD:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE people_list [
  <!ELEMENT people_list (person*)>
  <!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT birthdate (#PCDATA)>
  <!ELEMENT gender (#PCDATA)>
  <!ELEMENT socialsecuritynumber (#PCDATA)>
]>
```

```
<people_list>
  <person>
    <name>Fred Bloggs</name>
    <birthdate>2008-11-27</birthdate>
    <gender>Male</gender>
  </person>
</people_list>
```

From <https://en.wikipedia.org/wiki/Document_type_definition>

2.2.3 JSON

Javascript object notation (JSON) is a textual based programming language independent data format, designed to exchange data between applications. It is specified as an ISO/IEC standard (ISO/IEC 21778:2017 Information technology — The JSON data interchange syntax).

JSON format is more lightweight than XML but just as easy for humans to read and write and for machines to parse and generate. The more compact notation uses fewer characters to encode the same data, it is thus more efficient at transmitting data over networks.

The format is based on unordered sets of name/value pairs. Names must be of type string, values can be string, number, "True", "False" or "Null", objects or arrays. The format however does not include information what the type of each element should be and there is a certain ambiguity if how a number should be interpreted (type int, float or other).

A way to validate and restrict the types of data in a JSON format is to use a schema that specifies the types as well as additional restrictions for the values of the objects and variables. The schema needs to be present at validation. Including a validation step in the process however uses more computation time for decoding data.

JSON does not support comments in the data files.

As this format is also text based, it is not as compact as a binary format can be. The text needs to be encoded using Unicode UTF-8.

Typical syntax:

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
```

```
 "spouse": null
}
```

From <https://en.wikipedia.org/wiki/JSON>

2.2.4 YAML

YAML is another text-based data interchange format. It is more data oriented than XML and the latest version accepts JSON files as valid. It uses Python style syntax, defining blocks by indentation but can also use flow style with '[' and ']' or '{' and'}' to describe the data structure. Unlike JSON it supports comments. The text needs to be encoded using Unicode character sets.

YAML is an open format that is specified openly (https://yaml.org/spec/1.2/spec.html). It is however not fixed in a standard by an international body or association.

YAML includes JSON as a subset offers however further features mainly pertaining to more complex data structures (relational anchors, extensible data types or mappings preserving key order to name a few). However, for the purpose investigated here these features do not add significantly to the usefulness of YAML.

Another drawback with YAML is that it has a rather open syntax that lets the same data be represented in several ways creating different sizes of serialized data and needs a more complex encoder or decoder.

Typical syntax example:

```
---
receipt:     Oz-Ware Purchase Invoice
date:        2012-08-06
customer:
    first_name:   Dorothy
    family_name:  Gale

items:
    - part_no:   A4786
      descrip:   Water Bucket (Filled)
      price:     1.47
      quantity:  4

    - part_no:   E1628
      descrip:   High Heeled "Ruby" Slippers
      size:      8
      price:     133.7
      quantity:  1

bill-to:  &id001
    street: |
            123 Tornado Alley
            Suite 16
    city:   East Centerville
    state:  KS

ship-to:  *id001

specialDelivery:  >
    Follow the Yellow Brick
    Road to the Emerald City.
    Pay no attention to the
    man behind the curtain.
...
```

Source: https://en.wikipedia.org/wiki/YAML

2.2.5 EXI

Efficient XML interchange language is a binary format that tries to make XML more efficient. It is the binary XML encoding recommended and supported by the World Wide Web Consortium (W3C) and is specified as a W3C recommendation at https://www.w3.org/TR/exi/. It is equivalent to XML at the information set layer (will generate the same information with same structure than XML). Being binary it can reduce the verbosity of XML and with it the size of the serialized data. It also reduces parsing costs. To improve the performance further schemas can be included giving the algorithm more information on the data, however the schema must be present during serialization as well as during deserialization.

2.2.6 CBOR

Concise binary object representation (CBOR) is a binary data serialization format. It is loosely based on JSON and works using name/value pairs. The format being binary is not easily readable by humans, can however be much more efficient than text-based formats.

Data type information for major types: integers, byte string, text string, array, map, tag of number N, simple/float (length is specified, indefinite length possible) date/time strings are supported.

The format was designed to support encoding and decoding on constrained nodes (according to RFC 7228), as well as for high volume transfers. Thus, the formatted data is compact, and the encoder/decoders do not need a lot of computing or memory resources. It was developed to be used with internet of things devices which usually send a lot of data but have only constrained computing possibilities.

The format is specified by the Internet Engineering Task Force (IETF) in RFC 8949 [16].

A human readable diagnostic notation is specified which can be used during debugging.

2.2.7 CDR

Common data representation is developed by OMG (Object management group (also specify DDS protocol)) and part of OMG IDL. It is specified in this context by CORBA v3.0 (https://www.omg.org/cgi-bin/doc?formal/02-06-51). It is almost exclusively used within the CORBA environment.

This format is a binary format and thus not human readable. It assumes prior agreement on type and does not include information about types in the data representation. The OMG interface description language is used to define the data. This makes it lightweight as it includes just the data in a binary format.

An extended version of CDR, that supports evolvable types, is used within the DDS (distributed data service) middleware.

2.2.8 OPC UA Binary

The OPC UA Binary is a data format developed by the OPC foundation to meet the needs of OPC UA applications. According to OPC UA specification [15], the format is designed primarily for fast encoding and decoding while also considering the size of the encoded data on the wire. It is well integrated into the OPC UA environment and uses a broad range of primitive types including Booleans, Integers, Floating Point, String, DateTime, ByteString and several more specialised types. The format is almost exclusively used within the OPC UA framework.

Next to the binary format, OPC UA also defines OPC UA XML and OPC UA JSON formats for data serialization to be compatible with XML or JSON based applications over the web. All the data formats are specified in OPC 10000-6 Part 6: Mappings.

2.2.9 Apache Thrift

Apache Thrift does not define the serialization per se but rather an interface at program (application layer) level and forms a remote procedure call (RPC) framework more lightweight than CORBA or SOAP (XML based) as it is kept simple and uses binary.

Structure of data is defined using interface description language. This description is used for serialization and deserialization.

Several predefined serialization protocols are included in the framework: binary, compact binary, http-friendly (over JSON).

The implementation of Apache Thrift is based on the white-paper: https://thrift.apache.org/static/files/thrift-20070401.pdf. Apache Thrift is not standardized but open source (Apache license 2.0) and maintained by Apache Foundation. A wide range of standard APIs are available.

Depending on the serialization used the data can be human readable (however it will then take up more space). The mandatory use of interface descriptions however makes for human friendly handling of the serialization. However, as the data serialization is thus not self-describing, a copy of the interface description needs to be present on all nodes that need to deserialize the data.

2.2.10 Protocol buffers

Protocol buffers is a framework similar to Apache Thrift for distributed applications to communicate and exchange data. It is developed and maintained by google under an open source license. Serialization uses binary types (not human readable, an ASCII version is implemented for debugging purposes but is not backwards nor forwards compatible). It is designed to be smaller and faster than XML.

Comes natively with Code generators for C++, Python, Java, C#, Go, Ruby, Javascript… (protobuf 3.0), other languages have third party implementations (C, Perl, …).

As Apache thrift an interface description language is used to define a schema for serialization. The serialization is not self-describing and needs a copy of the interface description for the deserialization of the data.

Reference: https://developers.google.com/protocol-buffers

2.2.11 Apache Avro

Apache Avro is a data serialization system that provides similar functionality to Apache Thrift or Protocol Buffers. It serializes data in a binary format based on a schema. It can also use JSON to encode the data although this is mainly intended for debugging purposes.

The schema is stored with the data in a file. The data and schema are fully self-describing facilitating thus the data can be processed in a more dynamic manner than in Apache Thrift or Protocol Buffers, which need to generate code from the interface description prior to execution. Another advantage with this system is that if the program expects a different schema, as both schemas are present this can be easily resolved (missing fields, extra fields, etc.). Avro schemas are defined with JSON. Avro comes with optimization possibilities where schemas can be exchanged and retained at the beginning of a connection between two nodes. Thus, data can then be sent without repeating the schema at every transmission.

Avro has official releases for C, C++, C#, Java, PHP, Python, and Ruby

Apache Avro is available under an Apache License 2.0 (permissive free software license). (https://avro.apache.org/docs/current/)

2.2.12 ASN.1

The abstract syntax notation one (ASN.1), is a standard interface description language for defining data structures. Together with a set of encoding rules ASN.1 can be used to serialize and deserialize the data. Implementations of the standard in the form of compilers exist that create libraries of code from the ASN.1 data description, that can encode or decode data. These tools are well established for Java, C and C++.

The standard was created by the international telecommunications union (ITU) and is specified at https://www.itu.int/rec/T-REC-X.680/en.

Different encoding rules generate different outputs so the performance of this serialization varies depending on the encoding rules (binary: BER, DER, PER,… Human readable: XER, JER,…). Custom encoding rules can also be defined using the standardized encoding control notation (ECN) which is part of the ASN.1 family of standards.

The IEC 61375-2-3 [11] standard defines data structures using a system based on ASN.1.

2.3 Evaluation of data formats:

The evaluation of the relevant criteria is shown in Table 3 where the different serialization formats are evaluated

for relevant criteria using a qualitative scale (+: good, 0: neutral, -: bad). The following criteria were considered, although only the relevant ones are shown in Table 3.

- Readability by humans: Text based formats, although also encoded to bytes as UTF-8 can be directly read by most computers and programs and are thus considered human readable. It is very easy to read the data from these. All binary formats are not human readable and thus scored with a bad score for this criterium. Environments like Apache Thrift or Protocol Buffers support several encodings, including text-based ones. However, we always consider the binary format for these, as they will minimise the size of the serialized data and as the text based formats are mainly supported for development and debugging purposes and are not always forwards and backwards compatible.
- Data typing: This criterium is used to differentiate between formats that include data types in the data serialization (+), those who use a schema or interface description language to define the data types separate from the data (0) and those who do not support data type information (-).
- Performance of encoder/decoder: The performance of encoders and decoders is evaluated regarding computation/memory needs as well as speed. The exact performances are difficult to estimate as one would need to run benchmarks for each format with typical data, as some formats could be faster for certain types but not for others for example.
- Space needs of serialized data (directly linked to networking speed): The memory size of the serialized data is evaluated here. This is also the space the data takes up on the network when sent. To ensure fast communication as well as high network throughput, the size of the serialized data should be kept small. Here binary formats are almost always better than text-based formats. However, the size of the serialized data can also depend on the actual data and would need benchmarks with typical data to be evaluated definitively.
- Platform / language independence: Serialization data formats are developed with platform independence in mind. All data formats are platform and language independent. Thus, this criterium is not figured in Table 3.
- Availability of implementations: Some data formats are more widely used than others. This generally translates to more implementations of decoders and encoders in a more diverse set of languages. The implementations can be openly available or commercial products, usually with higher performance or more development tools. Here we also evaluate if the data format is widely used in relevant industry areas (automation, networking,…)
- Open / proprietary: None of the data formats is proprietary. However, we distinguish here whether the data serialization format is specified or standardised by an international institution and widely used (+), the format is specified or standardised by a non-international foundation but still widely used in industrial automation (0) or whether the format is standardized by an international institution or foundation but used by few key players on the market (-). The evaluation reflects the level of trust placed in the data format for its future relevance and continued maintenance of the standard. The Bitstream format was evaluated with a good mark (++) for this criterion, even though it could be considered not very open or even proprietary from the outside. However, as the format is completely controlled and specified within the system the concerns this evaluation criteria addresses are not relevant.
- Flexibility for upgrades: Here the rigidity of the data format to updates in the data, like adding new variables, is evaluated. A flexible format needs only small changes to the applications to handle a change of the data.
- Complexity of supported data structures: Some data formats can handle more complex data structures than others, as for instance dictionaries, references to other objects or arrays of mixed typed elements. However, as the needed data complexity is rather low for this application, all data formats can support sufficiently complex data. Therefore, this criterion will not be evaluated further.

	Readability by humans	Data typing	Performance of encoder/ decoder	Size of serialized data	Availability of APIs	Open/proprietary	Flexibility for upgrades
Bitstream	-	-	++	++	-	++	-
XML	+	+	-	--	++	+	+
JSON	+	0	+	-	++	+	+
YAML	+	0	+	-	0	-	+
EXI	-	+	+	+	+	+	+
CBOR	-	+	+	+	+	+	+
CDR	-	+	+	+	+	0	0
OPC UA Binary	-	+	+	++	+	0	0
Apache Thrift	-	0	+	++	++	-	0
Protocol buffers	-	0	+	++	++	-	0
Apache Avro	-	0	+	+	+	-	+
ASN.1	-	+	+	++	+	+	0

Table 3: Comparison of different data serialization formats.

There is no data format that excels in all the criteria and is an obvious pick. Further, some criteria are more important in some use cases than others. Thus, the need to differentiate between different use cases arises. On one hand we will consider process data for time-sensitive applications going over TSN-Ethernet such as the vehicle locator data for instance. On the other hand, we will consider message data for non-time-sensitive applications going over standard Ethernet, like diagnostic messages for instance. Several data formats can coexist on the CCN that are tailored to the needs of the applications. It is however preferable to keep the data formats somewhat uniform throughout the applications to make for a more modular and coherent system. This will facilitate application development.

In general however, due to the long life-time of the CCN, it is preferable to have a solution that can either be completely controlled by the specifications of the CCN or that is specified in a standard by an international organization or widely used in the industry of interest such that unforeseen changes can be prevented, Thus we will not consider formats with a bad rating in the open/proprietary criterium.

2.3.1 Time critical applications

For time critical applications a fast encoding and decoding is needed to meet the strong timing requirements. Also, the size of serialized data is important due to the limitation of the maximum payload of an (TSN-)Ethernet frame of 1500 bytes. Other criteria like readability by humans, data typing, and availability of APIs are less important. This implicit weighting of the criteria is considered in the following listed possible formats for time critical applications.

2.3.1.1 Bitstream

For process data for applications relying on fast data transmission or even real-time, it is essential to have small, serialized data sizes as well as fast encoding and decoding of the data. The perfect example for this is bitstream. They sacrifice readability and flexibility for smaller data sizes and fast encoding as the encoder can be specifically written for the data it works on.

2.3.1.2 OPC UA Binary

Another format that is also used in industry for real time applications is OPC UA Binary. OPC defines not only a serialization format but also a data exchange protocol that can work with TSN in order to deliver real time process data. It is well established in industry and supported by several key players in automation such as ABB Automation, Siemens, B&R industrial automation, Bosch Rexroth, etc. It is a modern solution that is however still evolving and due to the breadth of services provided, not all implementations are compatible. It would integrate well with the OPC UA communication protocols, is however seldom used outside of this framework.

2.3.1.3 CBOR

CBOR being a binary, self-describing language that can be encoded and decoded using limited resources could be an interesting alternative in case of constrained nodes such as embedded systems for instance. It will still be relatively short as it uses a binary format but will not manage to compete with a bit-stream or a schema informed language.

2.3.1.4 ASN.1

ASN.1 using BER or PER is another suitable standard that is quite widespread due to its early standardization and use in telecom industries. However, most of the implementations are in-house developments or commercial products. Only a few open implementations exist. The syntax of the description language is well known and used in standards relevant to the railway industry. (IEC 61375-2-3 [11] for instance).

2.3.1.5 CDR

When using the DDS middleware at the session layer, extended CDR together with OMG-IDL is used by DDS. To get to the full potential of DDS and its data-centred approach, it would be counterproductive to use another format.

2.3.1.6 Other

Other formats for that are fully schema informed could be used like Apache Thrift, Protocol Buffers or Apache Avro. As they are fully schema informed, they also have small sizes of serialized data. From these Protocol Buffers would probably be the best choice as it is well established (comparded to Avro which relatively new) and well documented (compared to Thrift which has less documentation). However, Protocol Buffers were not made to be used in embedded environments (although stripped down implementations exist, (nanopb)) and are a less universal standard as they are not standardised by an international body of standardisation. They are maintained and specified by google which is a company not necessarily aligned with the railway industry.

2.3.2 Non-time-critical applications

For non-time-critical data such as diagnostic data, a human readable data format would be well suited. Especially for maintenance purposes the debugging and reading of messages could be made a lot easier as the data can be accessed directly and in a comprehensible manner. Moreover, the widespread use of certain text-based data formats also in other industry fields as well as their good standardization, means that they will continue to be relevant in the future. Other criteria like fast encoding and decoding and size of serialized data are less important. This implicit weighting of the criteria is considered in the following listed possible formats for time critical applications.

2.3.2.1 JSON

Due to its small size and good performance, JSON would be the most suitable solution. For explicitly introducing the data types of the data fields, the use of JSON schemas would be preferable.

2.3.2.2 XML

XML is verbose, and the size of the serialized data is higher than JSON. Even though, as XML is already used for different standardized communications, it is also solution for non-time critical applications.

2.3.2.3 others

YAML as the last of the text-based formats is not standardized by a reputable standardization body and has a very open syntax that would allow for the same data to be represented in too many ways, making the size of the format less predictable and not improving the readability of the data. Therefore, it is not proposed as a preferable solution.

2.3.3 Conclusion

Considering the differentiation between time critical and non-time critical data for the evaluation, the following possible formats remain for the corresponding applications.

Possible formats for time-ciritical application data over TSN-Ethernet	**Possible formats for non-time-ciritical application data over standard Ethernet**
Bitstream	JSON
OPC UA Binary	XML
CBOR	
ASN.1	

Table 4: Possible Data Serialization Formats considering the respective application

For safety- and time-critical CCS-applications todays interfaces specifications are defining bitstream packets. Thus, it is not necessary to have a self-describing format and one can expect all the applications to follow the specified interfaces. And with reserved parts in the data packets there is still flexibility for further updates. Therefore, Bitstream is recommended for the use of time-ciritical applications.

For non-time-critical applications JSON with the use of schemas is recommended considering the evaluation of the different criteria.

This is however only a recommendation. Several data formats can coexist on the same bus and others could be used. E.g. CBOR can be used to have a short binary format that uses few computational resources for time-critical data. Or OPC UA Binary could be integrated into applications making use of the OPC UA communication protocol on session layer. XML can still be used too, where it is already used (e.g. UIC 559). The aim of this recommendation is to create an ecosystem on the CCN that is as uniform as possible.

As already mentioned, when discussing the OPC UA binary format, the recommendation might be influenced by the choice of session layer protocol. At the moment, TRDP 2.0, OPC-UA PubSub and DDS/RTPS are still considered. The session layer protocols OPC-UA and DDS also provide or integrate, in the case of DDS, the data encoding using OPC UA binary or extended CDR.

3 Network Architecture and Cybersecurity

3.1 Overview

3.1.1 Network Architecture Gamma Release [10]

The assessment of different network architectures during gamma phase concluded for new trains in long term vision that CCS and TCMS network shall be combined on the same physical network but logically segmented into different virtual networks (VLANs). It was proposed to integrate the CCN in ECN as separate virtual networks. This solution ensures a good hard-real-time behaviour while having a clear logical separation between the CCS and TCMS domains. This solution is shown in the following figure.

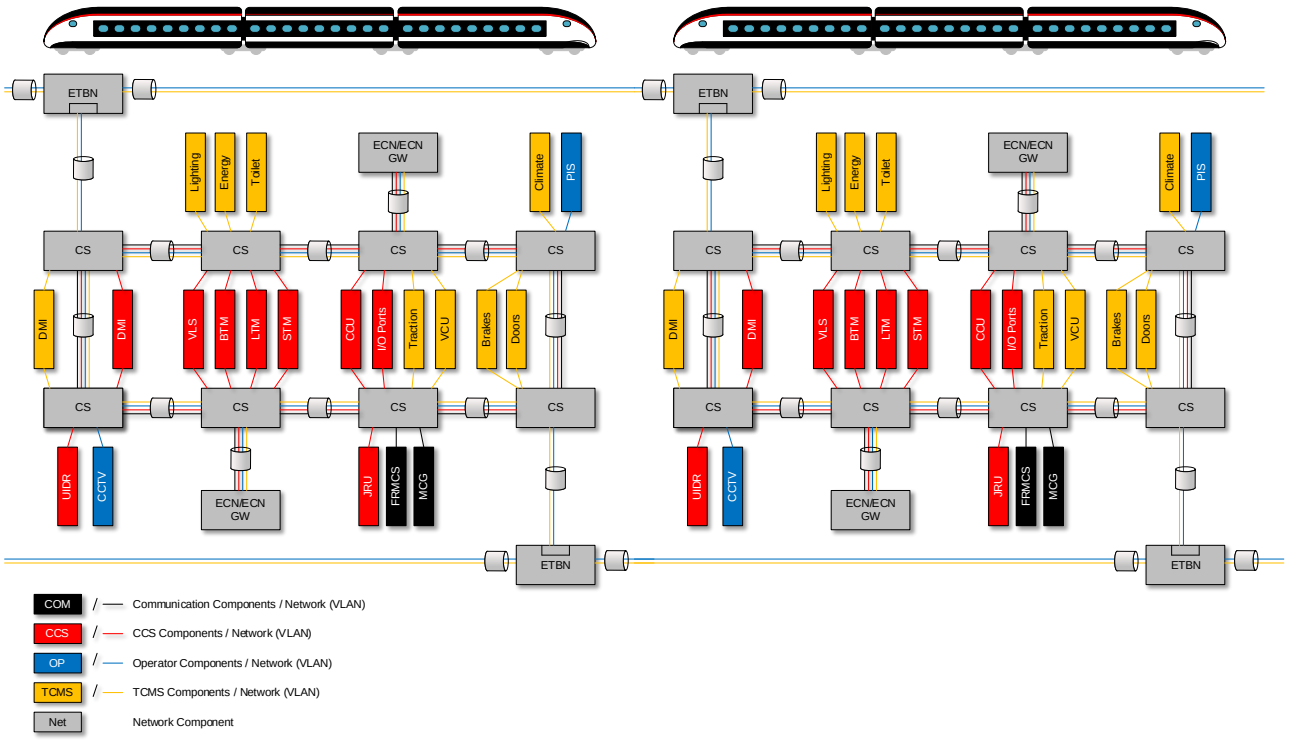

Figure 2: Physical network architecture with CCN integrated in ECN with logical separation by VLANs

3.1.2 IEC 62443-3-3 [12] and prTS 50701 [13]

In the industry sector the standard series IEC 62443 is established for cybersecurity. The railway sector adopted this standard series and shows in the preliminary technical specification prTS 50701 how to apply the industry standard IEC 62443. Based on a threat analysis, followed by a risk analysis the relevant Security Level will be derived. For CCS applications the security level will most likely be defined as SL3. In the following table the protection corresponding to a specific security level is defined.

Security Level	Protection against attacker type
SL1	Protection against casual or coincidental violation
SL2	Protection against intentional violation using simple means with few resources, generic skills and a low degree of motivation
SL3	Protection against intentional violation using sophisticated means with moderate resources, IACS specific skills and a moderate degree of motivation
SL4	Protection against intentional violation using sophisticated means with extended resources, IACS specific skills and a high degree of motivation

Table 5: Security Levels

In the IEC 62443-3-3 standard requirements for different security levels are defined. Regarding network topology for CCN the following requirements on the restricted data flow are important:

Security Level	System Security Requirement 5.1 and enhancements
SL1	The control system shall provide the capability to logically segment control system networks from non-control system networks and to logically segment critical control system networks from other control system networks.
SL2	The control system shall provide the capability to physically segment control system networks from non-control system networks and to physically segment critical control system networks from non-critical control system networks.
SL3	The control system shall have the capability to provide network services to control system networks, critical or otherwise, without a connection to non-control system networks.
SL4	The control system shall provide the capability to logically and physically isolate critical control system networks from non-critical control system networks.

Table 6: System Security Requirements 5.1 – Network segmentation from IEC 62443-3-3

All system security requirements from IEC 62443-3-3 are generally applicable to railway applications according to the security levels (SL-T) of the zones and conduits in the system under consideration (SuC). Nevertheless, due to the peculiarity of the railway application, the prTS 50701 informs about the existence of railway specific considerations as guidance. To the system security requirement SR 5.1 the following railway notes are listed.

Security Level	Title	Railway notes
SL1	Network segmentation	In response to an incident, it may be necessary to break the connections between different network segments. In that event, the services necessary to support essential operations should be maintained in such a way that the devices can continue to operate properly and/or shutdown in an orderly manner. This may require that some servers may need to be duplicated on the control system network to support normal network features, for example dynamic host configuration protocol (DHCP), domain name service (DNS) or local CAs. It may also mean that some critical control systems and safety-related systems be designed from the beginning to be completely isolated from other networks.
SL2	Physical network segmentation	Independence from non-control networks is required at SL2. In case physical segregation is technically not feasible or even an increase in cybersecurity risks, a logical segregation concept is acceptable explicitly if the following associated system security requirements [SR 1.2, SR 1.8, SR 1.9, SR 3.1/SR 3.1 RE 1 , SR 3.7, SR 4.1/SR 4.1 RE 1, , SR 6.2, SR 1.5 RE 1] are fulfilled.
SL3	Independence from non-railway application networks	
SL4	Logical and physical isolation of critical networks	The criticality of a railway application is determined by the risk assessment and that should influence the logical and physical isolation. The usage of segmentation methods like different fibres or colors for fibre-optic cables or the usage of cryptographic measures like those mentioned in EN 50159 are ways to implement this requirement in railway applications.

Table 7: System Security Requirement notes on SR 5.1

The system security requirement SR 5.1 for SL2 and higher from IEC 62443-3 require physical segmentation between control system networks and non-control system networks as well as between critical control system networks and non-critical control system networks. In the described network architecture in 3.1.1 the logically segmented zones are actually segmented in time (time division multiple access) over different predefined TSN streams for process data. If this solution can be treated as physically segmented, is not yet clear. Nevertheless, the prTS 50701 supports the clear physical segmentation between control system networks and non-control system networks. But it allows logical segmentation between critical and non-critical control system networks even on SL2 or SL3.

3.2 Outlook

Considering the requirements from IEC 62443-3-3 and the railway notes in prTS 50701 one physical network (even though physically segmented in time) for control system networks like CCN or ECN and non-control system networks like operator network will not be acceptable from a cybersecurity point of view. Therefore, the control system networks shall be physically segmented from the non-control system networks. Non-control system networks can be operator networks for e.g. CCTV, passenger information. Passenger networks for public internet access or entertainment on passenger devices must be physically segment as well from control networks like CCN or ECN. Also, the communication devices for the access to the trackside systems shall be

physically separated. This leads to the following proposal on possible network architecture.

The zoning concept of the network architecture proposal was derived from the functions and its criticality. Therefore, the two different zones for CCN/ECN with logical segmentation are defined:

1. Red Zone: Critical control network for CCS and critical TCMS components (e.g. traction, brakes, doors)
2. Green Zone: Non-critical control network for auxiliary functions (e.g. toilet, climate control, lighting)

Due to the criticality of the functions in the TCMS domain, it is proposed that the critical control functions traction, brakes and doors are put together with the CCS domain in the same red zone. Especially as soon as the automatic train operation (ATO) function is added to the train, the CCS domain and the traction, braking and doors functions are closely related.

Physical segmented from the CCN/ECN network and each other there are the following zones:

3. Black Zone: Communication network for mobile network link to trackside.
4. Blue Zone: Operator network for his own specific functions (e.g. CCTV, passenger information)

In the middle of all zones there is a ECN/ECN security gateway in between observing and controlling the data traffic crossing a zone boundary.

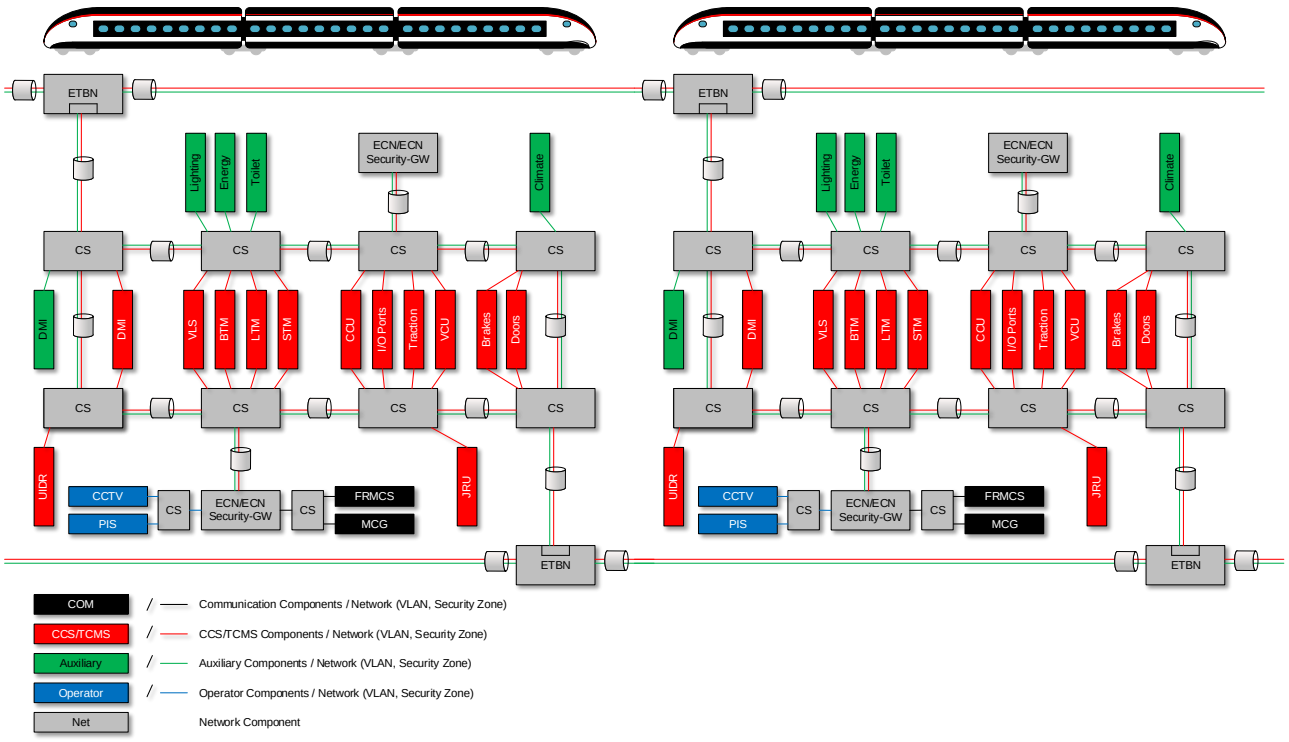

Figure 3: Proposal on possible physical and logical network architecture with CCN integrated in ECN

It must be considered that this proposal comes from a cybersecurity perspective of the CCN and must be discussed and investigated in the subsequent phases of the OCORA initiative. Furthermore, the CCS and TCMS domains must elaborate the same understanding of the network architecture. The future network architecture must be aligned with other programs like e.g. Shift2Rail with its projects CONNECTA or X2Rail or consortia like e.g. UNISIG or Unife. X2Rail-3 is currently investigating the cybersecurity of the “Drive-by-Data Architecture” of CONNECTA [14] until end of June 2021. Afterwards an alignment between all involved parties is needed. At the end, the results shall be incorporated into the next version of IEC 61375 standard.