802.1CBdb

Extended stream identification functions

Background

- Project scope:

  Define additional stream identification functions allowing the support of a wider set of applications requiring the QoS that TSN can provide.

  Additionally, address 802.1CB-2017's errors and clarifications.

- Focus (so far) = identification of non-TSN traffic

- Two proposals made for untagged traffics
  - Ethertype-based identification
    - Main application: adaptation of legacy industrial network protocols to TSN
      - Brownfield migration
  - Mask & match identification
    - Generic Layer-2 + upper-layers identification

A complement to the existing functions

ETHERTYPE-BASED IDENTIFICATION

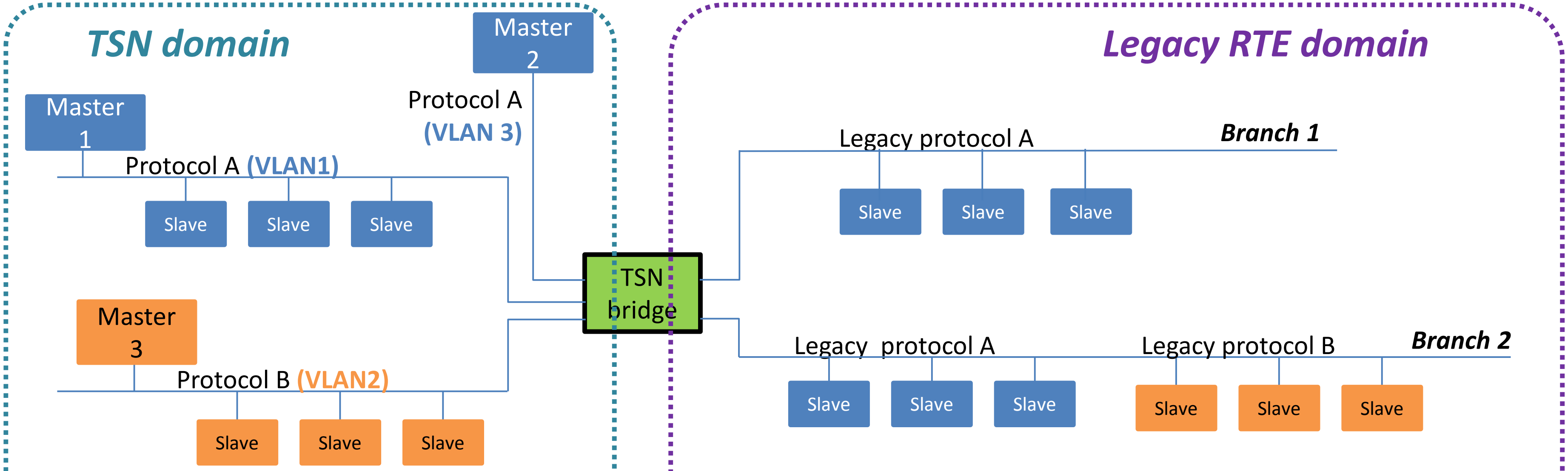
Background

- All (IEC) standardised Real-Time-Ethernet (RTE) industrial automation network protocols have a dedicated Ethertype code:

Ethernet IA protocol	EtherType
EtherNet/IP(DLR)	0x80E1
PROFINET	0x8892
EtherCAT	0x88A4
POWERLINK	0x88AB
SERCOSIII	0x88CD
CC-Link IE	0x890F
…	…

Use case

- Interdomain communication: brownfield migration



- Streams uniquely identified by: DestMAC + VLAN-ID

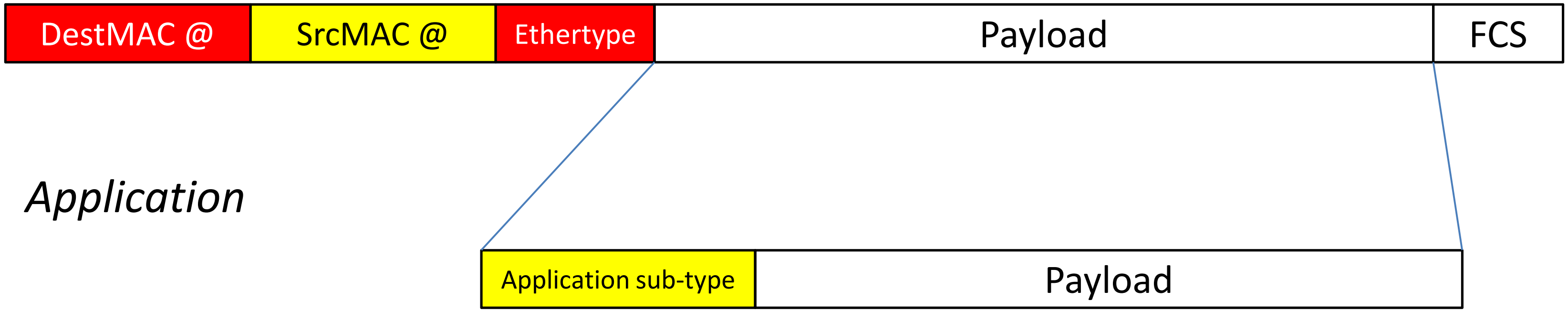
- Untagged traffics
- Legacy RTE protocol identified by a dedicated Ethertype
- Application (upper) layer provides additional information in the frame payload to distinguish between different traffic types: real-time, cyclic, acyclic, time sync, etc… -> Application sub-type

Inter-domain streams

- How to handle inter-domain streams
  - Legacy Slave to TSN Master stream (stream identification)
    - DestMAC necessary to address the right Master
    - Ethertype + Application sub-type can be used to further identify the stream (and its priority)
      - Define the nature of the stream: cyclic, acyclic, etc… -> Traffic Class
      - Ethertype can also be used to determine the right VLAN
    - SourceMAC can optionally be used to distinguish source slaves
      - When slaves on the same branch are individual stream sources
  - TSN Master to Legacy Slave stream
    - Stream id (DestMAC1 + VLAN-ID) -> destMAC2
      - Using active DestMAC + VLAN Down stream identification function
      - SourceMAC « copied » from TSN domain frames
      - Ethertype « copied » from TSN domain frames

Summary

- Stream identification parameters
  - *Ethernet*
  - *Application*



🟨 optional

🟥 mandatory

– The Length of the Application Sub-type field has to be determined

» 8 bits ?

A superset of the existing functions

MASK & MATCH-BASED IDENTIFICATION

Description

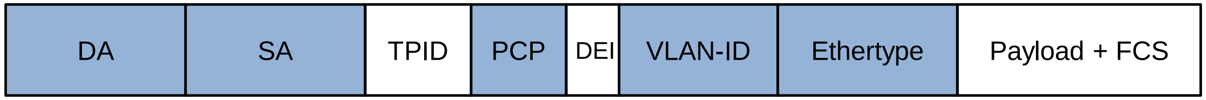
- A generic identification function
  - Superset of the existing identification functions
  - Based on a template

- Based on 2 types of identification parameters
  - *Layer-2* (*L2*) identification parameters
  - *Upper-layers* (*UL*) identification parameters

- Stream identification function = uses a combination of L2 and UL identification parameters

Solution proposal

- L2 identification parameters
  - Considering 1 level of VLAN encapsulation only, the parameters match the Ethernet frame header fields that can participate in the indication of stream content, and selection of path and queues in the stream’s frame forwarding operations
    - *Dest MAC*
    - *Source MAC*
    - *PCP*
    - *VLAN-ID*
    - *Ethertype*



DA	SA	TPID	PCP	DEI	VLAN-ID	Ethertype	Payload + FCS

  - 2 options to define (encode) an L2 parameter set
    - a unique value per valid header fields combination
    - a (5-bit) bitmap indicating the presence/absence of a given header field in the L2 parameter set.

Solution proposal

- UL identification parameters
  - Definition of a UL parameter list
    - 1 UL parameter = 1 protocol field defined by its:
      - *Offset* = distance from the beginning of the payload (assumption: protocol fields are byte-aligned)
      - *Length* = protocol field length in bits
    - Indication of the number of UL parameters
    - UL parameter list format:
      - {Nb UL parameters = N; [UL param 1], [UL param 2], ..., [UL param N]}
      - UL param n = (Offset,Length)

Solution proposal

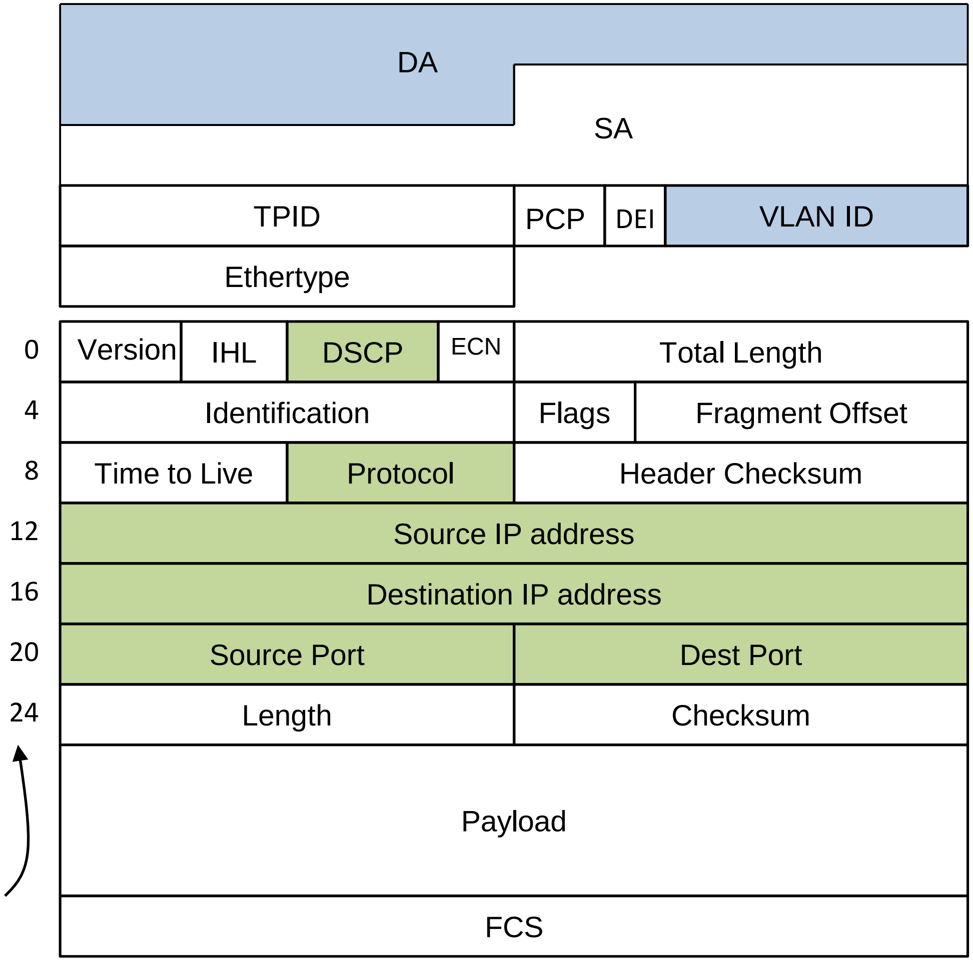
- An example:
  - Based on 802.1CB-2017’s IPv4 + UDP stream identification (using L2 parameters bitmap encoding)

- L2 parameter bitmap

  {1,0,0,1,0}

- UL parameter list

  {6; /* Nb UL param */
  (1,6), /* DSCP field */
  (9,8), /* Protocol */
  (12,32), /* Source IP */
  (16,32), /* Dest IP */
  (20,16), /* Source Port */
  (22,16)} /* Dest Port */



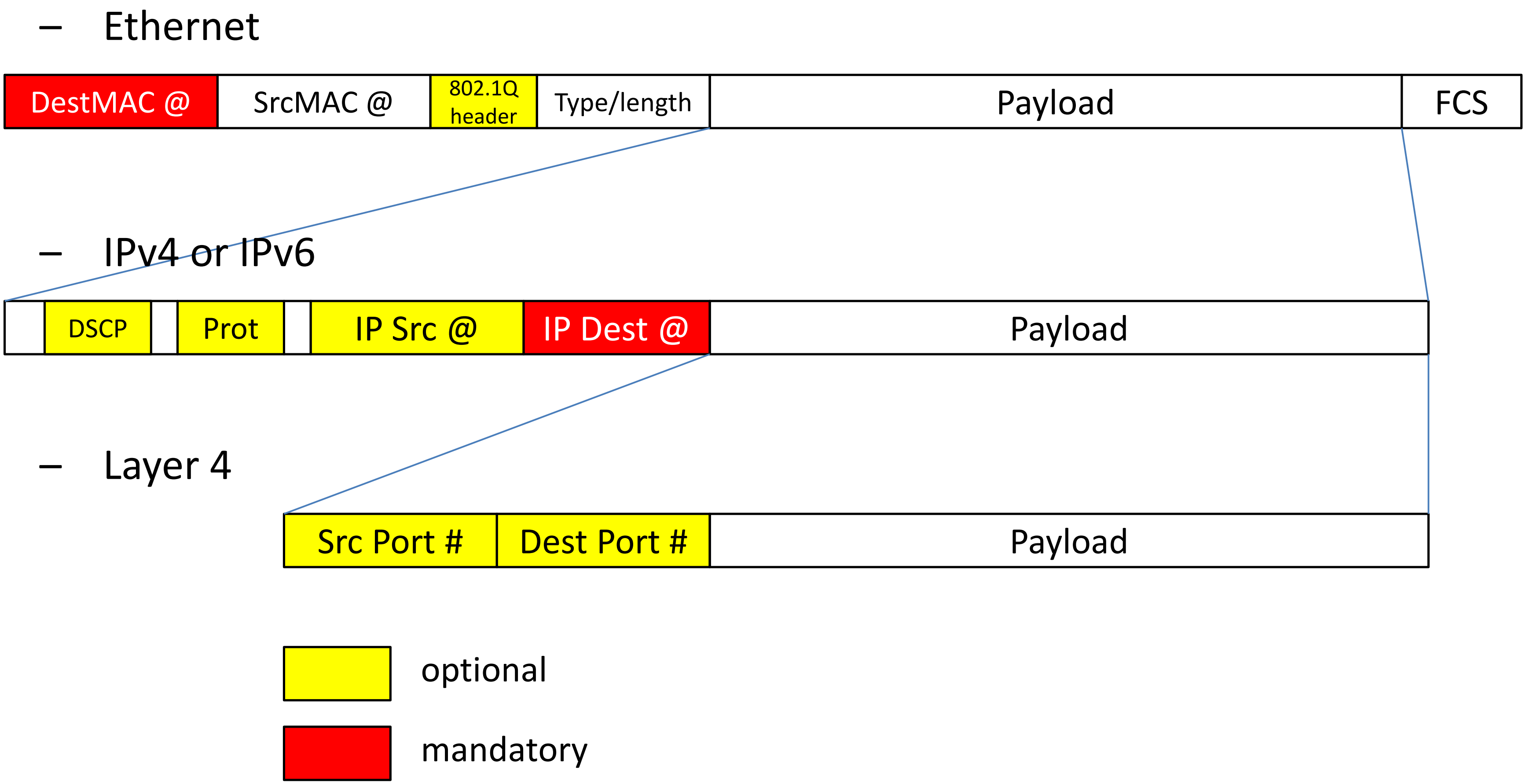
Byte offset in the Ethernet payload

Clarification to 802.1CB-2017

IP STREAM IDENTIFICATION ISSUE

IP stream identification in 802.1CB-2017

- Mandatory and optional fields for IP stream identification in 802.1CB-2017



  - Ethernet

DestMAC @	SrcMAC @	802.1Q header	Type/length	Payload	FCS

  - IPv4 or IPv6

	DSCP		Prot		IP Src @	IP Dest @	Payload

  - Layer 4

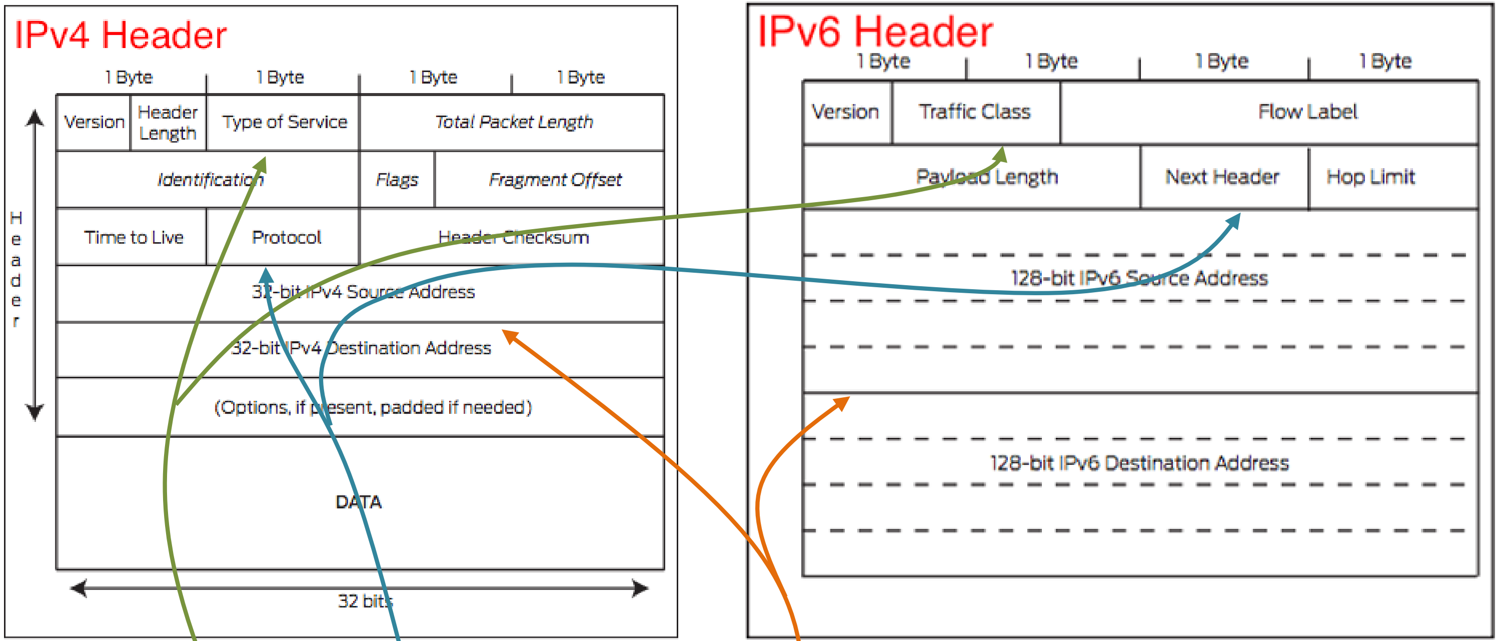
Src Port #	Dest Port #	Payload

🟨 optional

🟥 mandatory

Issue

- IPv4/IPv6 distinction



IPv4 Header

1 Byte | 1 Byte | 1 Byte | 1 Byte

Version | Header Length | Type of Service | *Total Packet Length*

Identification | *Flags* | *Fragment Offset*

Time to Live | Protocol | Header Checksum

32-bit IPv4 Source Address

32-bit IPv4 Destination Address

(Options, if present, padded if needed)

DATA

IPv6 Header

1 Byte | 1 Byte | 1 Byte | 1 Byte

Version | Traffic Class | Flow Label

Payload Length | Next Header | Hop Limit

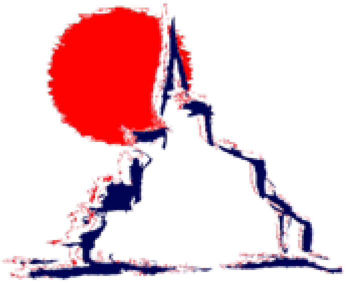
128-bit IPv6 Source Address

128-bit IPv6 Destination Address

- IPv4 and v6 packets have different payload layouts
  - DSCP, transport protocol id, IP addresses and port numbers are located at different offsets
- Do we need to add the *Version* field to the managed objects in 802.1CB to provide a mean to distinguish between IP versions or,
- Do we consider that the DestMAC (and optionally the VLAN-ID) are enough to determine the IP version ?

Comments for discussion

- Ethertype
  - The “application sub-type” is an optional field
    - This information is present in CC-Link IE, PROFINET and SERCOS III, but is it the case in other RTEs (e.g. EtherCAT) ?
  - Can we limit the “application sub-type” field to 1 byte ?
    - Ok for CC-Link IE, PROFINET and SERCOS III, but for other RTEs ?
- Mask&match
  - We can make the offset bit-aligned so that specific bit fields (not necessary byte-aligned) can be tested more easily
  - Do we need to define a wild card (“don’t care”) for the match operation ?
    - it is not defined in the existing identification function (for the MAC@, VLAN-ID, etc…) and can be considered as an implementation option.

Thank you for your attention 

MITSUBISHI
ELECTRIC

Changes for the Better