Layering for the TSN Layer 2 Data Plane

Norman Finn
Version 4

Mar. 3, 2014

Moving ahead TSN L2/L3 work

- At the AVnu face-to-face meeting in December, Norm Finn made presentations on the subject of ensuring that the IEEE 802.1 Time-Sensitive Networking (TSN) efforts are compatible with the established body of work from Internet Engineering Task Force (IETF) on the Internet Protocol (IP).
- A four-step plan for advancing the TSN work was presented.

Moving ahead TSN L2/L3 work

A. Pick at least one data plane model for joint L2/L3 **circuit identification**.

B. Pick at least one data plane model for joint L2/L3 **duplicate packet deletion**.

C. Pick at least one data plane model for the **dual-homed end station**.

D. Pick exactly one (hopefully) suite of **protocols** to fit the data plane choices made.

This presentation

- This is tsn-nfinn-L2-Data-Plane-0214-v04. It offers an improved model for layering the AVB/TSN concepts, which leads to a set of choices for answering the data plane questions A, B, and C. Contents of this presentation:
  - What's the problem? What's the fix?
  - Sublayer structure for TSN
  - Alternative TSN Encapsulations
  - Summary
- This presentation does not make choices, and so cannot advance to Question D, protocols.

NOTE

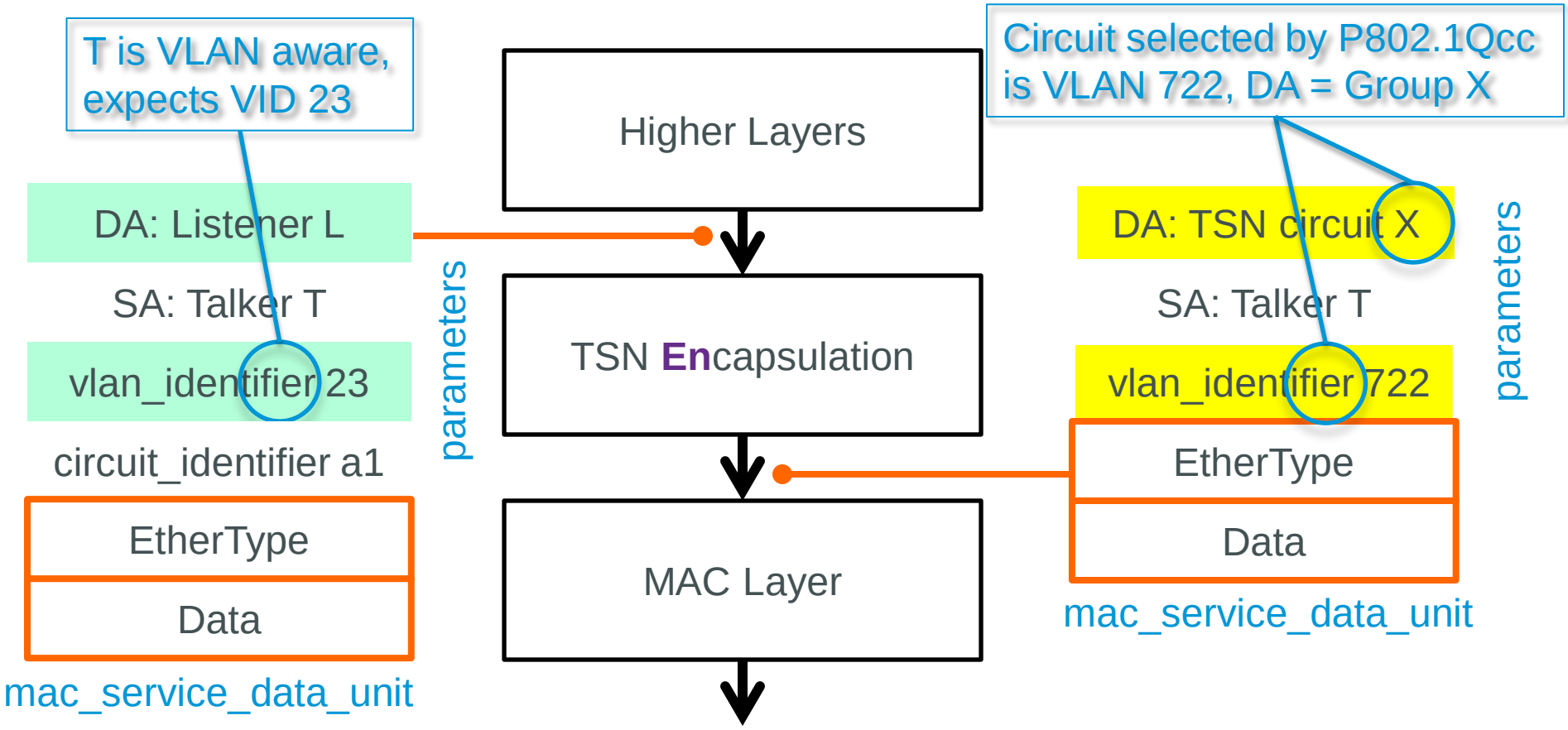
- Previous versions of this presentation,tsn-nfinn-L2-Data-Plane-0114-v02, etc. had the order of the layers wrong.

TSN Sublayers

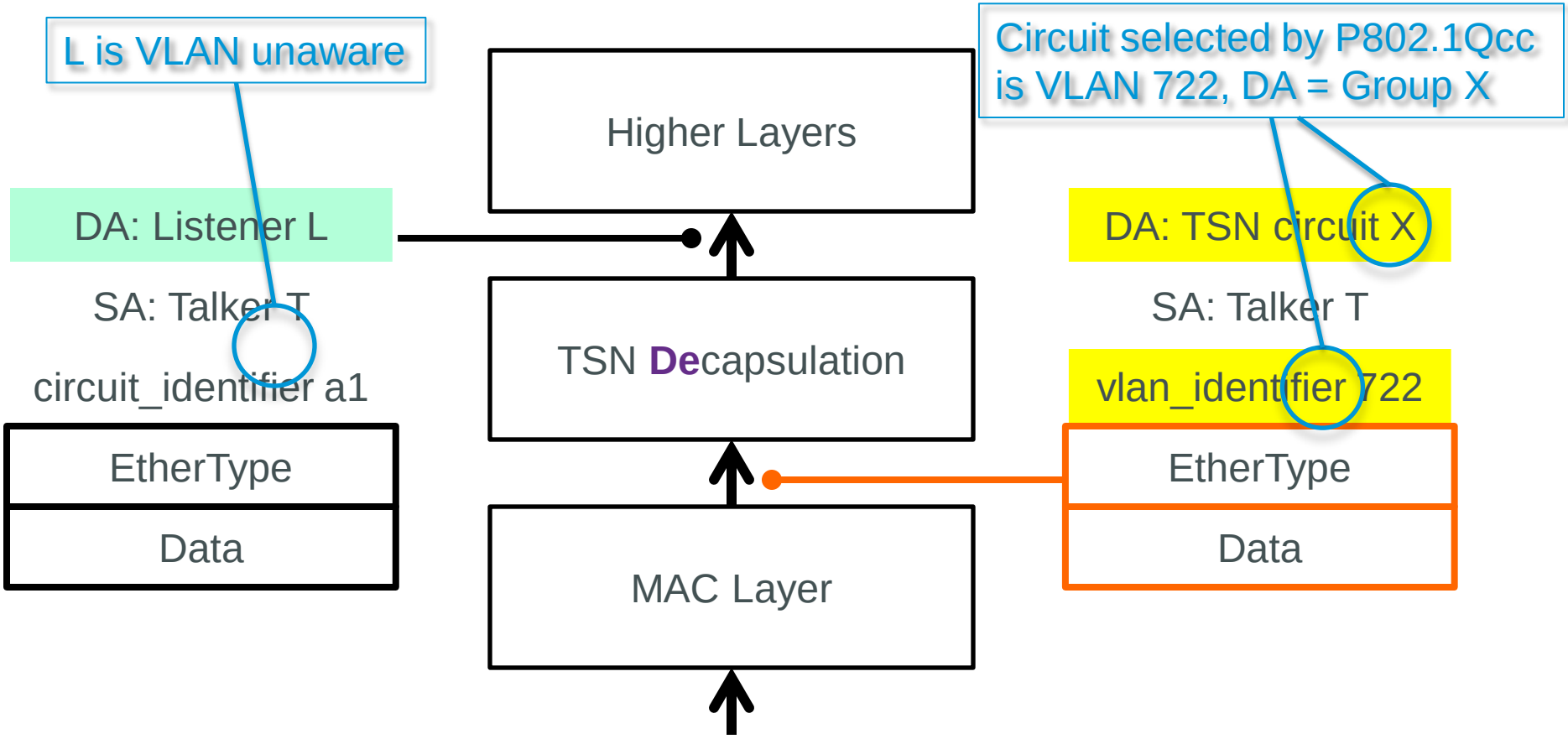
The problem?

- It's difficult to integrate an IP stack with TSN.
  - I want unicast IP streams to acquire TSN capabilities. They don't have Group DAs.
  - My IP stack uses VLANs in the conventional way; they are not assigned per-stream.
- Using normal VLANs, it is very difficult to arrange fixed (pinned down) paths.
- Using normal VLANs, MAC address learning gets confused by pinned down paths.

Quick summary for today's TSN: Correct stack in **Talker T**



Quick summary for today's TSN: Correct stack in **Listener L**



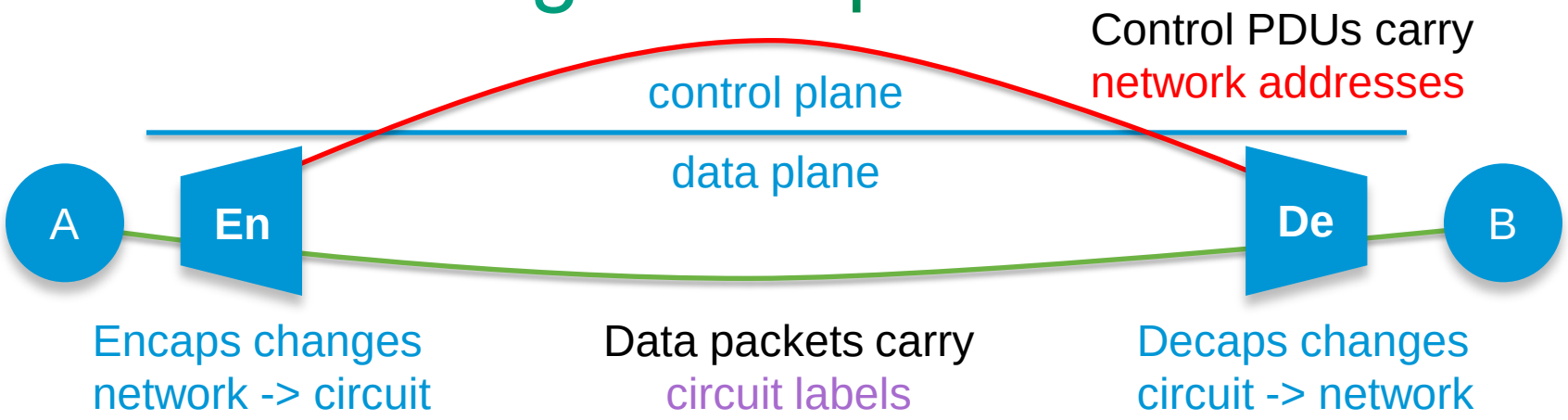
How does this get setup?

- **T**'s control functions use P802.1Qcc to say, "I want a connection to the function reached by a connectionless packet (an Ethernet frame) with VLAN 23 and Destination MAC address **L**."
- P802.1Qcc picks the encapsulation {VID 722, Group Address X}.
- Along the path from Bridge to Bridge, each bridge makes whatever modifications to the end-point addresses inside the P802.1Qcc PDU that would be made to the network packet.
  - For example, VID translations or tag removal.

How does this get setup?

- The edge Bridge adjacent to Listener **L**, in particular, removes the VLAN tag from the P802.1Qcc request, and tells **L** about the encapsulation.
  - **L's TSN Encaps/Decaps function is VLAN-aware, even if the rest of B's protocol stack is not!**

How does this get setup?



- The information required by the TSN Encapsulation and Decapsulation functions came from the control plane.

The end result?

- No problem with IP, including unicast streams.
  - The transformations are transparent to the IP stack.
  - The IP stack works just like it always has with ARP, etc.
- No problem with any other protocol that knows about or requires particular MAC addresses or VLANs
  - The {VLAN, address} is restored as it comes up the receiving stack.
- No problem with fixed paths.
  - Every circuit has a multicast address. One or two VLANs can support all fixed-path circuits, even with VLAN-aware host stacks.
- No problem with MAC address learning.
  - Because fixed paths are on a VID that doesn't do learning.
- **No problem with backwards compatibility.**
  - **Existing AVB stations see the same encapsulation as always.**
  - **We'll craft P802.1Qcc to allow the Talker to supply the tunnel address.**

Why didn’t we see this, before?

- The circuit encapsulation was so trivial, we didn’t realize that we were doing circuits with labels.
- The circuit encapsulation is so trivial, you can implement an Ethernet-only application that uses the circuit label as a network address.

What do we do?

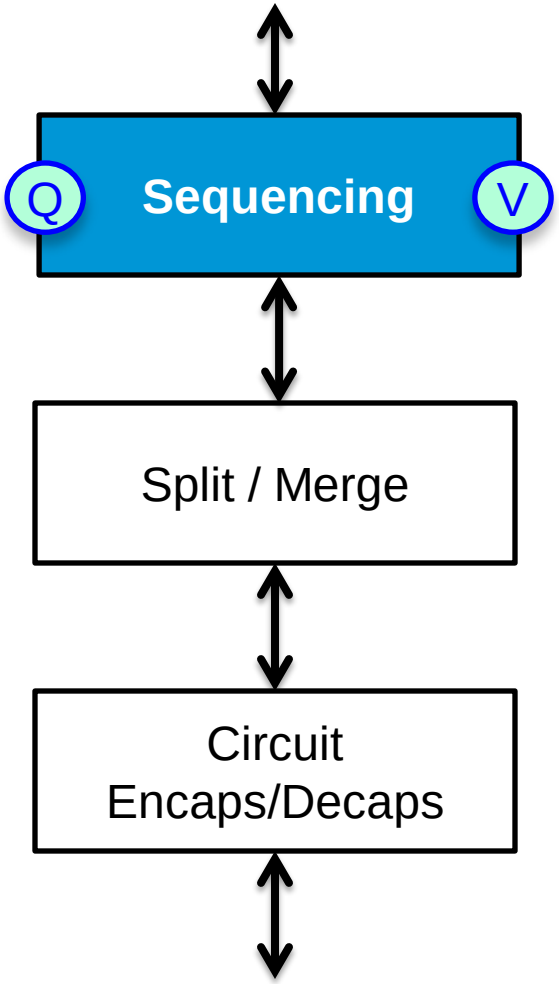
1. Revise 802.1Q to have the TSN Encaps/Decaps layer, working as described.
2. Fix P802.1Qcc to:
   - Take the connectionless target {VID, MAC address} pair as an input.
   - Modify that pair, as needed, as P802.1Qcc proceeds through the network.
   - Give the circuit {VID, MAC address} to the endpoints, rather that taking it as an input (except as necessary for legacy reasons).

Do I have to change my ASICS?

- If what you're doing now works, don't change it. We're not changing the bits on the wire.
- How or whether you do the full reconstitution of the frame inside your host stack is an implementation matter.
- If you want to provide a totally transparent service to the upper layers, you have the option of implementing the TSN Encaps/Decaps.
- Or, of implementing (or using your existing implementation of) several other protocols.

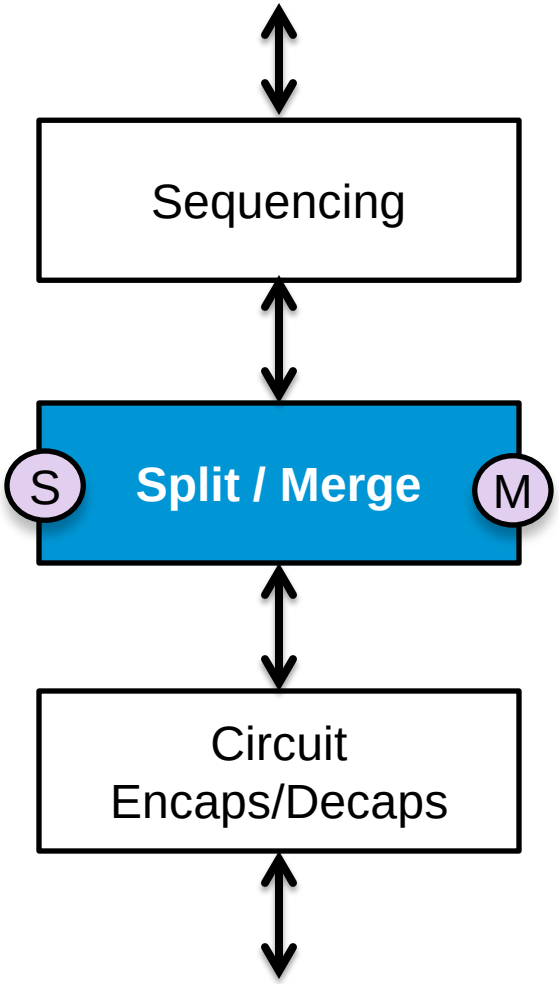
Layering

Functional elements required for TSN



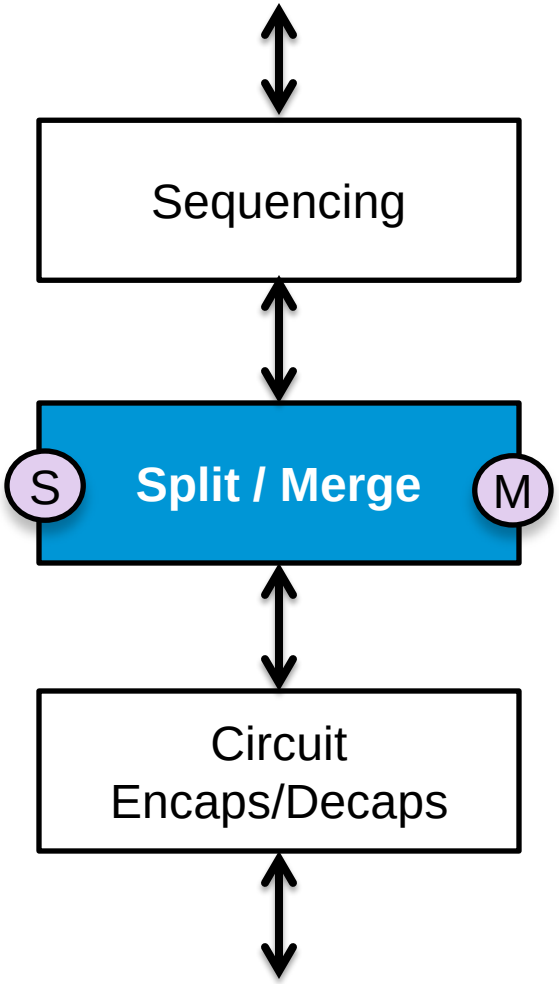
- **If** a circuit requires either seamless redundancy or long-range out-of-order delivery protection, then:
  - Packets are sequenced on transmit. (Q)
  - Duplicates are discarded on receive; out-of-order may be discarded. (V)
  - Packets may be buffered and reordered on receive.

Functional elements required for TSN



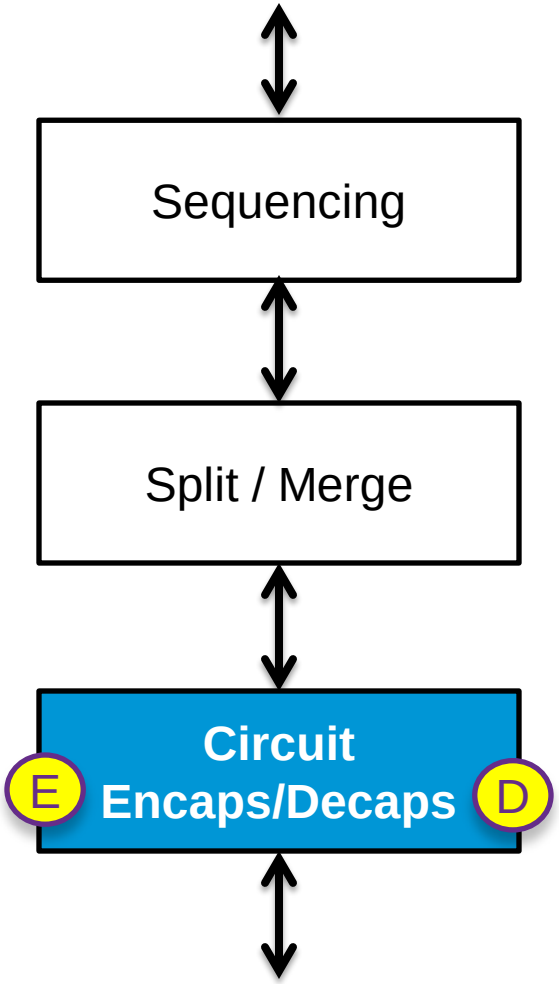
- **If** circuit requires multiple paths, differently labeled then its packets can be:
  - Replicated, (S) and relabeled, for transmit to one or more ports.
  - Merged into one port, (M) and relabeled, (but **duplicates are not discarded**) on receive.

Functional elements required for TSN



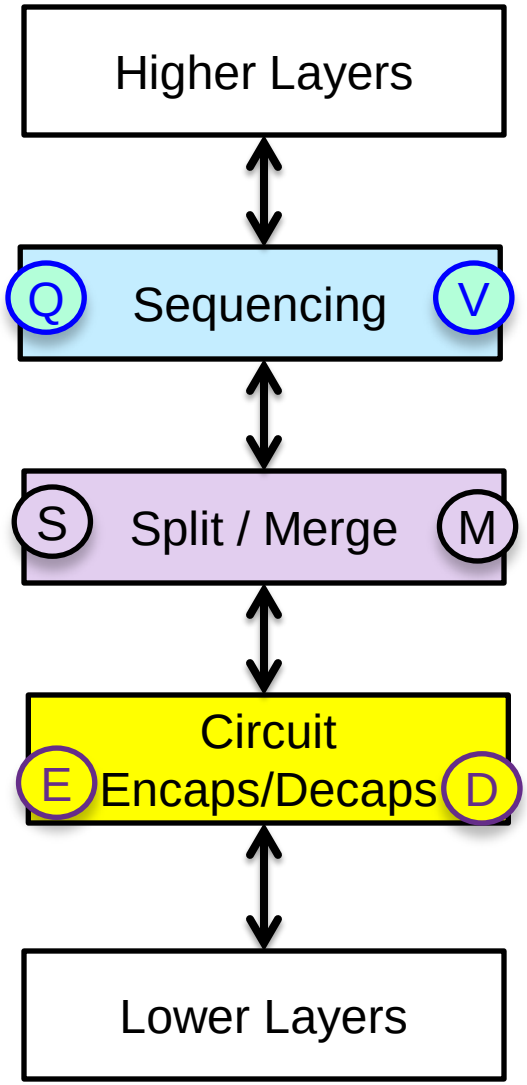
- Note that the Split Merge function does not have multiple physical ports.
- It replicates / merges the circuits, leaving it to normal networking to make the physical split.

Functional elements required for TSN



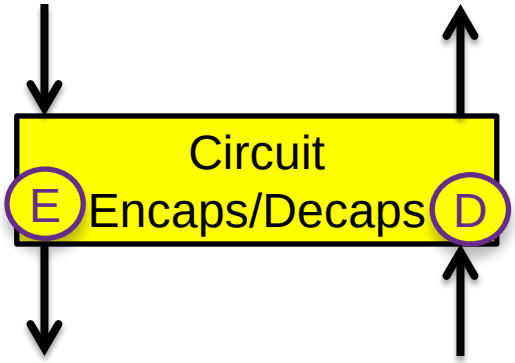
- Individual circuits must be identified, and packets encapsulated (E) and decapsulated (D), for:
  - Fixed paths;
  - Per-circuit resources;
  - Seamless redundancy.

Functional elements required for TSN



- The ordering of these sublayers is important, but they can be ordered in different ways to accomplish different purposes.

Circuit Encaps/Decaps sublayer

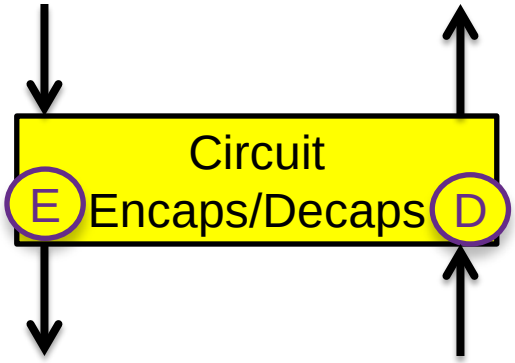


Addresses, circuit_identifier, sequence_number, Data

Addresses*, [circuit_identifier], [sequence_number], Data*

- The Circuit Encaps/Decaps sublayer has an interface on the upper SAP that includes at least a circuit_identifier parameter, and may include a sequence_number parameter.
- On the lower interface, these parameters are **encoded***, somehow, in the Addresses or in the Data, and can be parameters, as well.

Circuit Encaps/Decaps sublayer

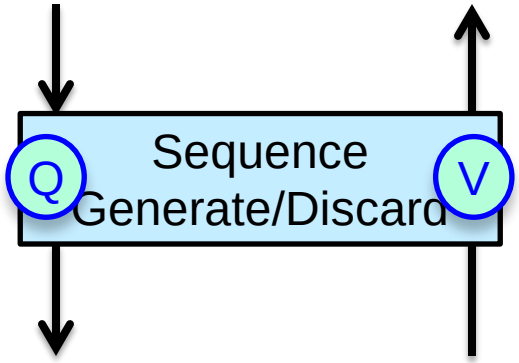


Addresses, circuit_identifier, sequence_number, Data

Addresses*, [circuit_identifier], [sequence_number], Data*

- Non-TSN data frames are assigned a circuit_identifier that allows them to pass through transparently, without any change.

Sequence Generate sublayer

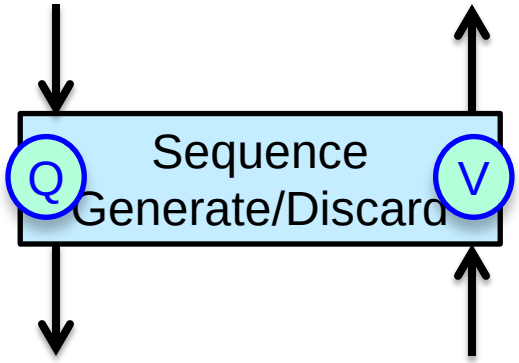


Addresses, Data, circuit_identifier, sequence_number

Addresses, Data, circuit_identifier, sequence_number

- The Sequence Generate sublayer (Q, only on the way down) creates sequence_number parameter values, usually sequential integers from 0 (or 1).
- (The sequence_number parameter input from the upper layers, if present, is ignored.)

Sequence Discard sublayer

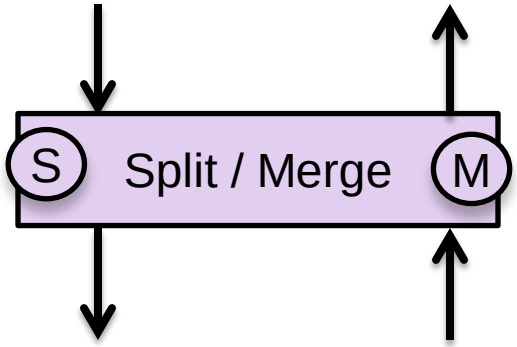


Addresses, Data, circuit_identifier, sequence_number

Addresses, Data, circuit_identifier, sequence_number

- The Squence Discard sublayer (V, only on the way up) is a shim, in that it has the same interface, top and bottom. That interface includes the circuit ID and sequence number.
- This shim **discards** out-of-order or duplicate packets, and **may buffer** and reorder them.

Split / Merge sublayer

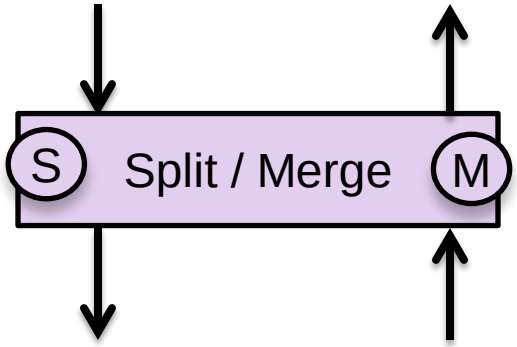


Addresses, Data, circuit_identifier, [sequence_number]

Addresses, Data, circuit_identifier, [sequence_number]

- The **Split** sublayer replicates packets received from the upper layers, and outputs them with circuit_identifiers that are different each other, and may be different than the input.
- The Split sublayer has only one lower port. There are other ways to describe it, but this seems to avoid confusion with the bridge and router relay and forwarding functions.

Split / Merge sublayer

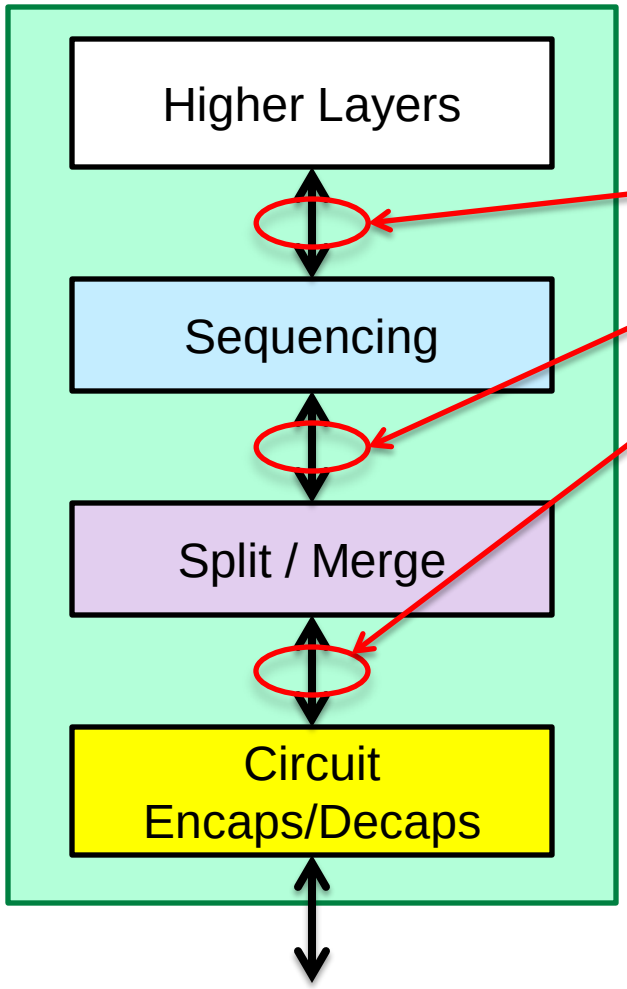


Addresses, Data, circuit_identifier, [sequence_number]

Addresses, Data, circuit_identifier, [sequence_number]

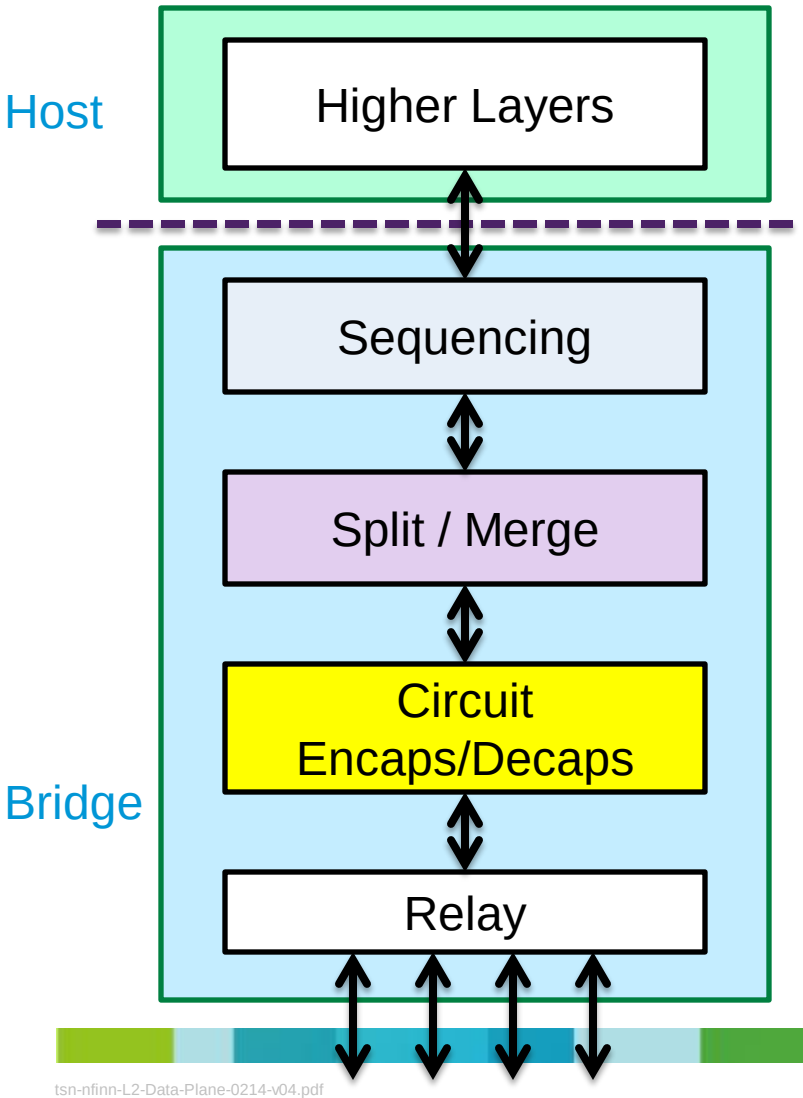
- The **Merge** sublayer translates the circuit_identifiers of the packets received on its one lower port, and outputs them on its upper port. **It does not discard frames**; it ignores the sequence_number parameter, if present. Discards are done by the Sequence Discard sublayer.

Single system



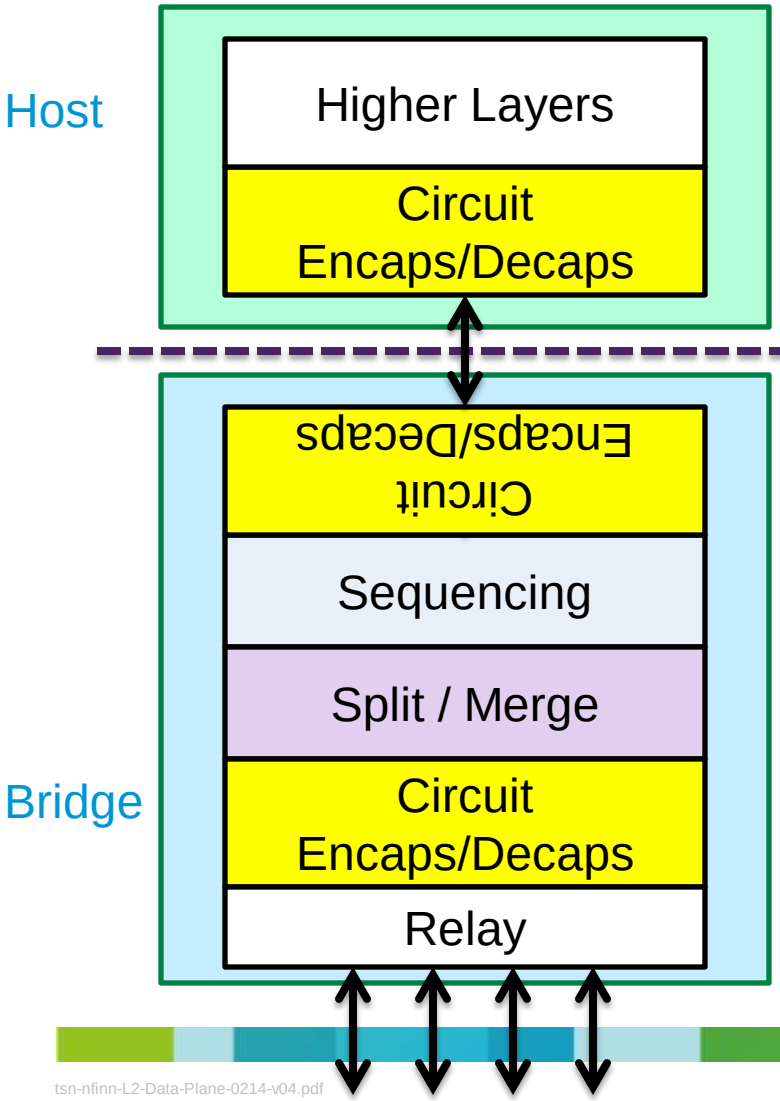
- Within a single system, the circuit_identifier can be **out**-of-band or **in**-band. Out-of-band methods include:
  - Socket ID.
  - Separate service instances.
  - Multiple Sequencing functions.

Multiple systems



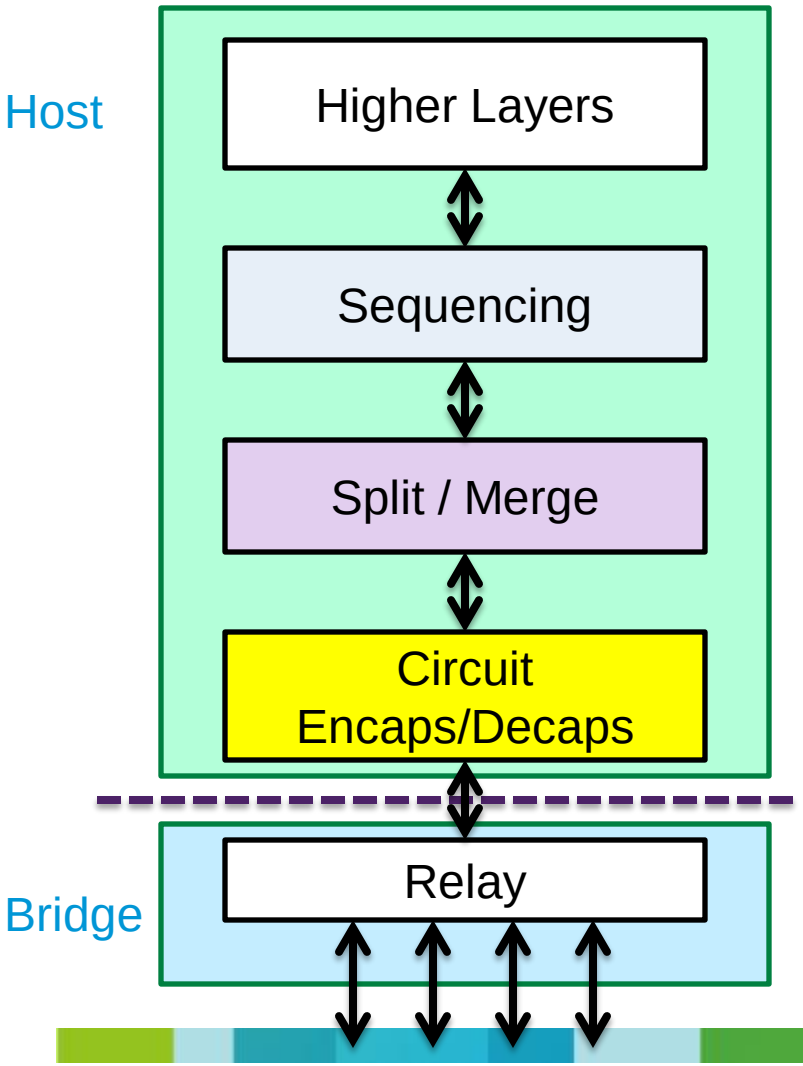
- For multiple systems, circuit identification must be **in-band**:
  - Some form of tag.
  - One or more layers of explicit addresses (e.g. VLAN ID or IP 5-tuple).
  - A circuit ID buried in an application.

Multiple systems



- This is one way to separate the functions in boxes.
- The Encaps/Decaps identifies the circuits, so the lower box (a network node) doesn't have to do deep packet inspection to identify them.

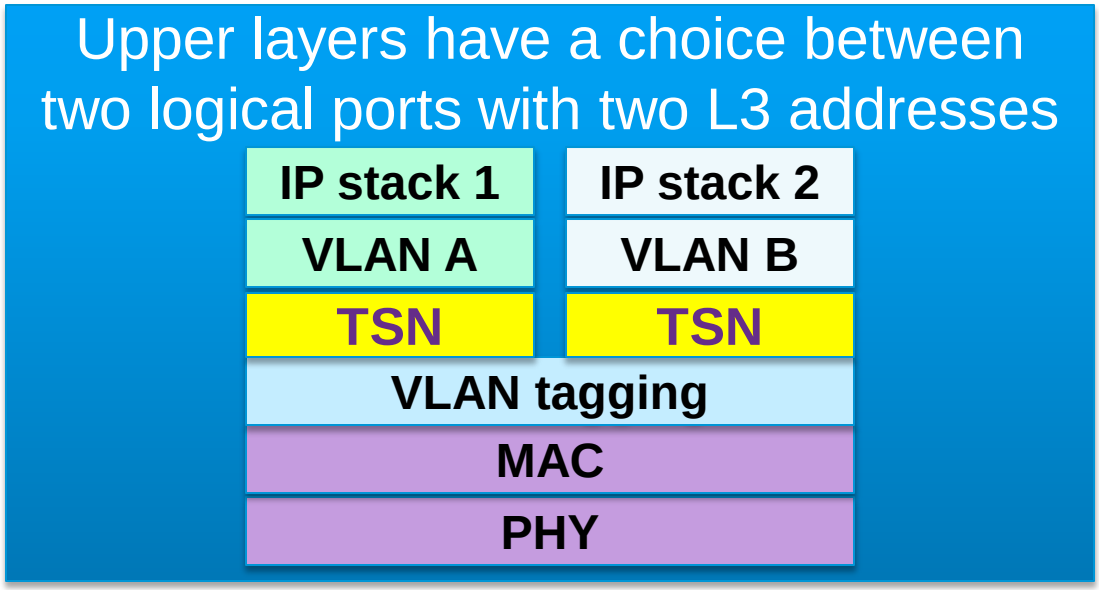
Multiple systems



- This is another way to separate the functions in boxes.
- The lower box (a network node) doesn't have to do deep packet inspection to identify the circuit.
- But, traffic is doubled across the link in (at least) the receive direction.

Single-port multiple VLAN host

- Common model for a **multi-VLAN host** with a single physical port (router or multi-VLAN server). One VLAN-aware (EISS) TSN stack would work, too.

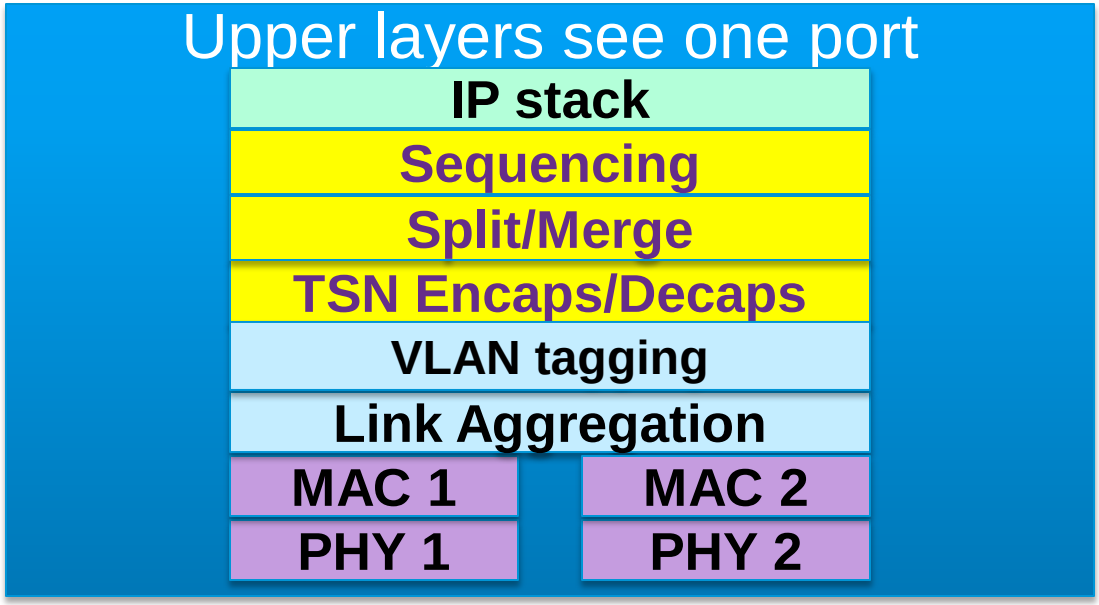


Single-port multiple VLAN host

- Even if the VLAN Tagging layer is not present (i.e., the host is VLAN-unaware), **the TSN Encaps/Decaps function is VLAN-aware**.
- No sequencing or split/merge functions were shown in the diagram. They could be present. Often, however, their functions would be proxied by the adjacent bridge, in which case the TSN MAC address provides a great circuit ID.

Dual-port non-relay host (Link Aggregation)

- The **DRNI** model works transparently with Seamless Redundancy. Link Aggregation splits regular traffic normally, and splits TSN traffic using the circuit ID or encapsulation.

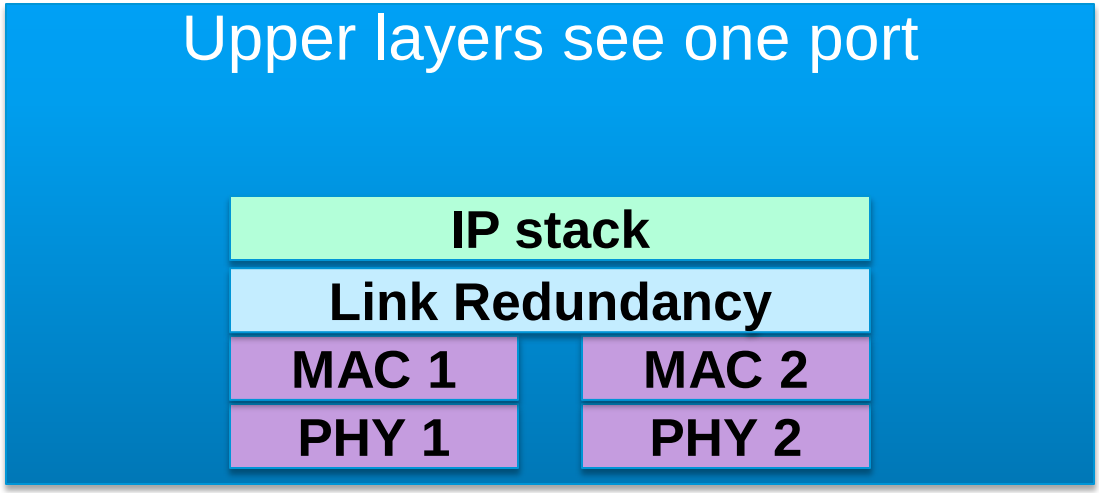


DRNI Host

- This diagram assumes that the Split/Merge function duplicates frames, and passes to the Link Aggregation layer (either explicitly in the frames, or by the circuit_identifier parameter) what it needs to split the flows.
- The DRNI host cannot be part of a ring – it can only be a dual-ported end station. (That's a use case, not a problem.)
- The non-TSN applications work just fine, without replication, because DRNI works.

Dual-port host (HSR or PRP)

- **IEC 62439-3 HSR/PRP** supports dual-homed hosts along with TSN. The Link Redundancy layer provides Sequencing and Split/Merge capabilities, within the limits of the HSR/PRP topology assumptions.

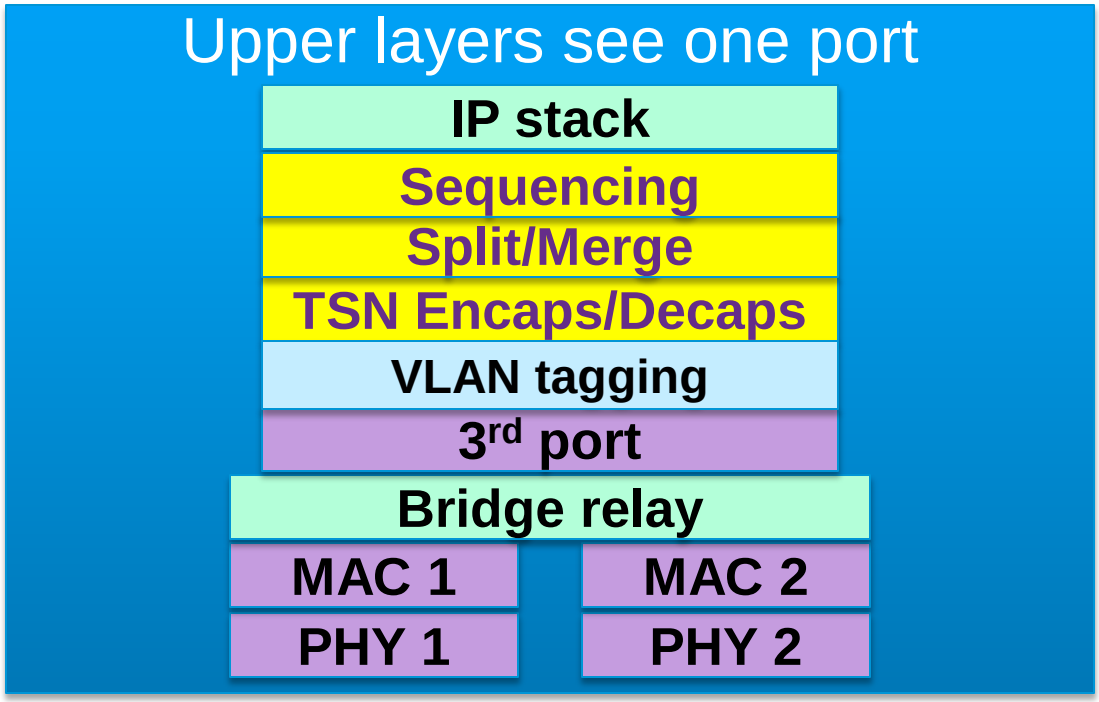


Dual-port host (HSR or PRP)

- As written, HSR supports a host that relays traffic from port to port, and PRP supports a host that does not.
- HSR requires a ring topology.
- PRP requires connections to separate networks.
- As we will see in the next section, both protocols can be adapted to work over a general purpose 802.1 network for TSN.

Dual-port relay host (bridged)

- The **Bridge** model works fine. The host stack creates **differently-labeled circuits**, and the bridge part directs them on different paths.

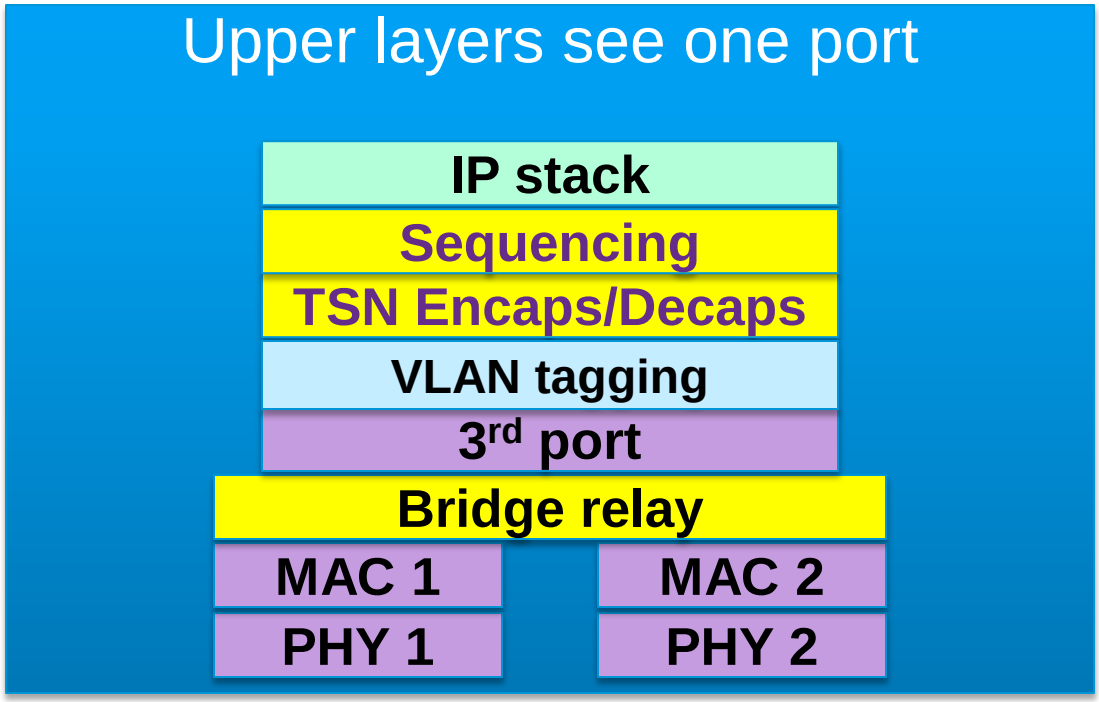


Dual-port relay host (bridged)

- In the preceding diagram, the Split/Merge function is assumed to alter the circuit_identifiers of the two streams, and generate two frames, each with a different encapsulation, so that the Bridge can send those frames on the other two Bridge Ports.

Dual-port relay host (bridged)

- The **Bridge** model works fine. The host stack creates **one circuit**, and the bridge just works. No split/merge layer needed.



Dual-port relay host (bridged)

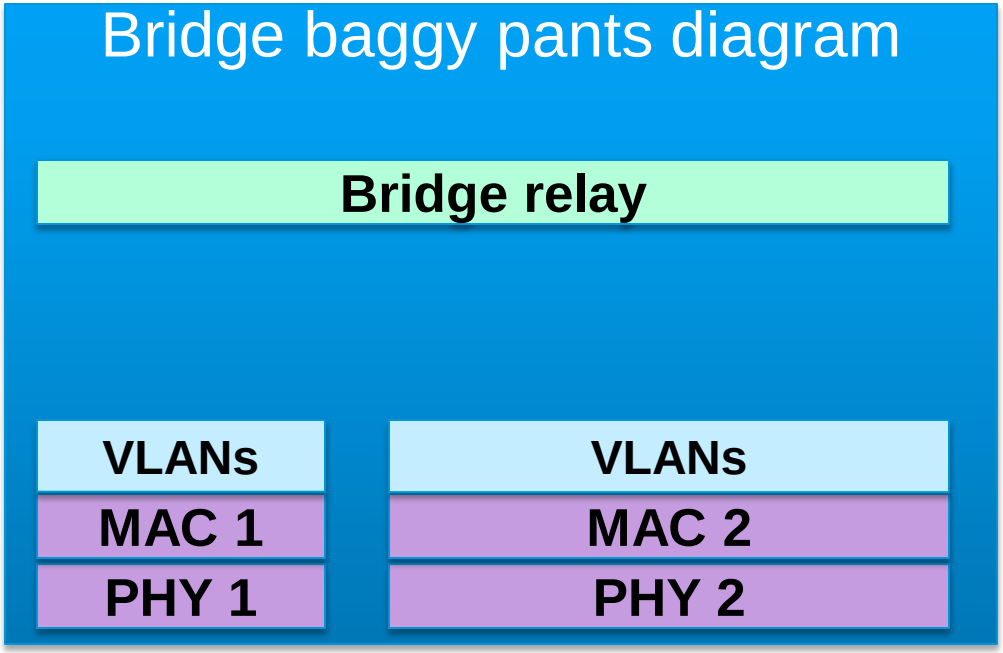
- In the preceding diagram, we do not change the circuit IDs in the host stack, and output a single frame (for each TSN circuit) to the Bridge.
- The standard Bridge Relay does its job, and no Split/Merge function is required.

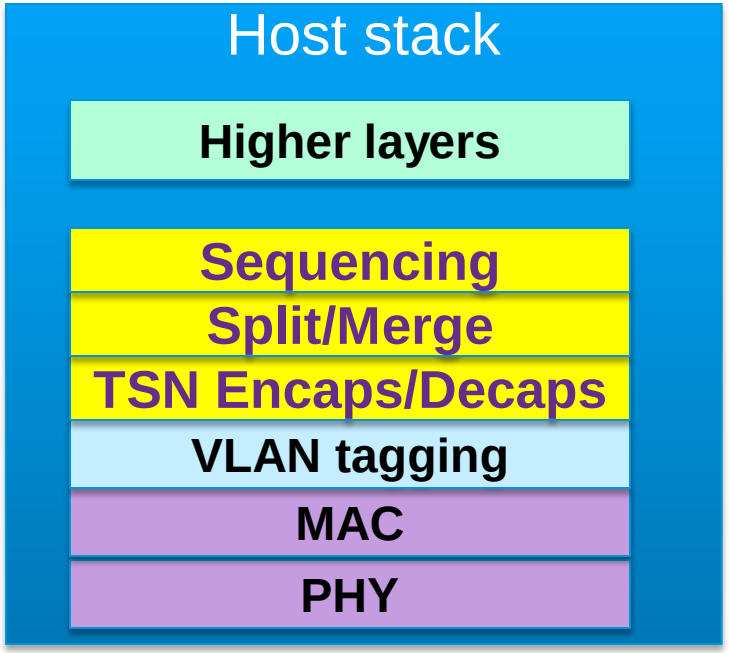
Dual-port relay host (bridged)

- This bridged models should be of particular interest to TSN, as they are the most general, and of course, 802.1 defines bridges.
- They automatically support rings or dual-homed stations.
- When combined with the Simple 2-port Intermediate System concept, this makes a powerful ring/chain node implementation.

Proxy bridge stack

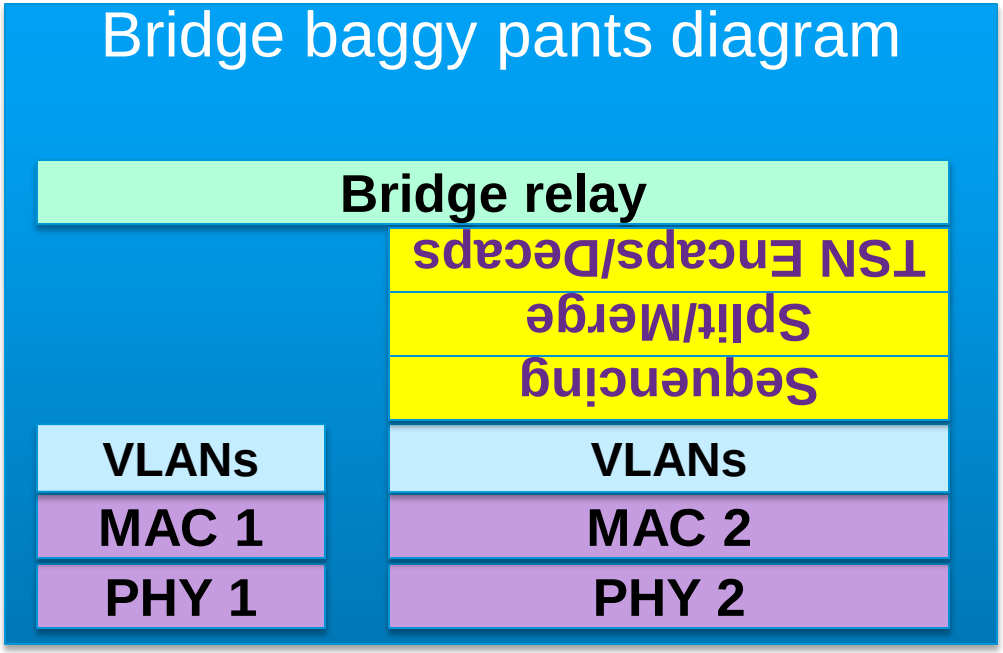
- A **Bridge** may want to offer proxy services to an end station that is TSN unaware.
- What you start with:

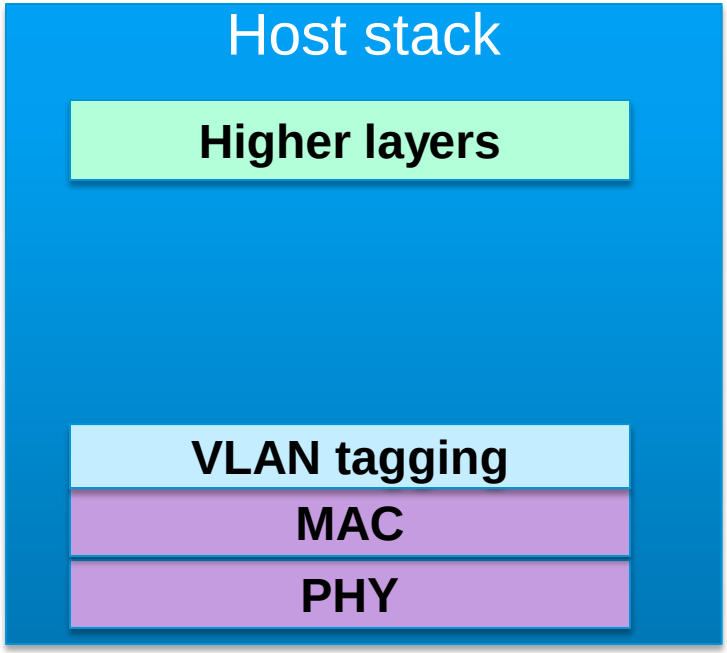




Proxy bridge stack

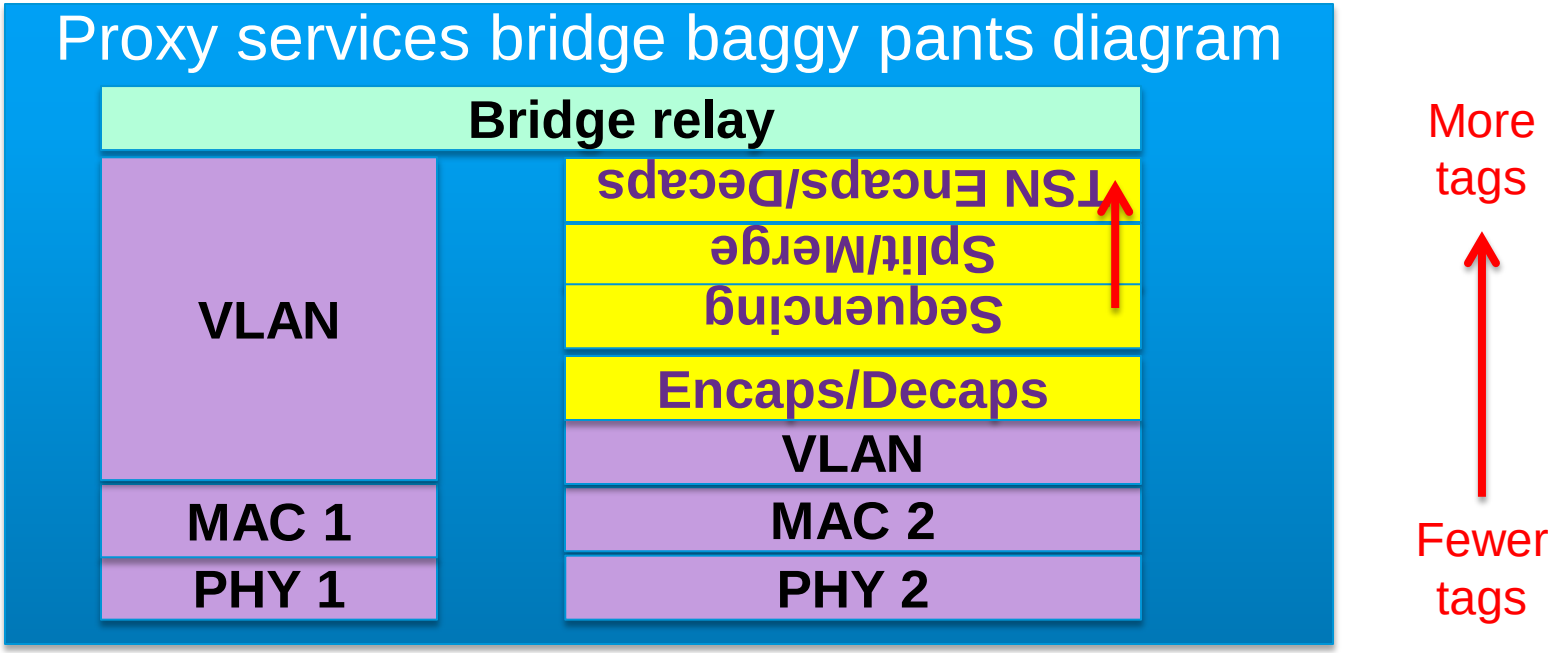
- Moving the Circuit Encaps/Decaps to the bridge means turning it upside down; the “clear” side is now below the “encaps” side.





Proxy bridge stack

- An extra Encaps/Decaps sublayer is then required to convert something explicitly in the data frame to a circuit_identifier:

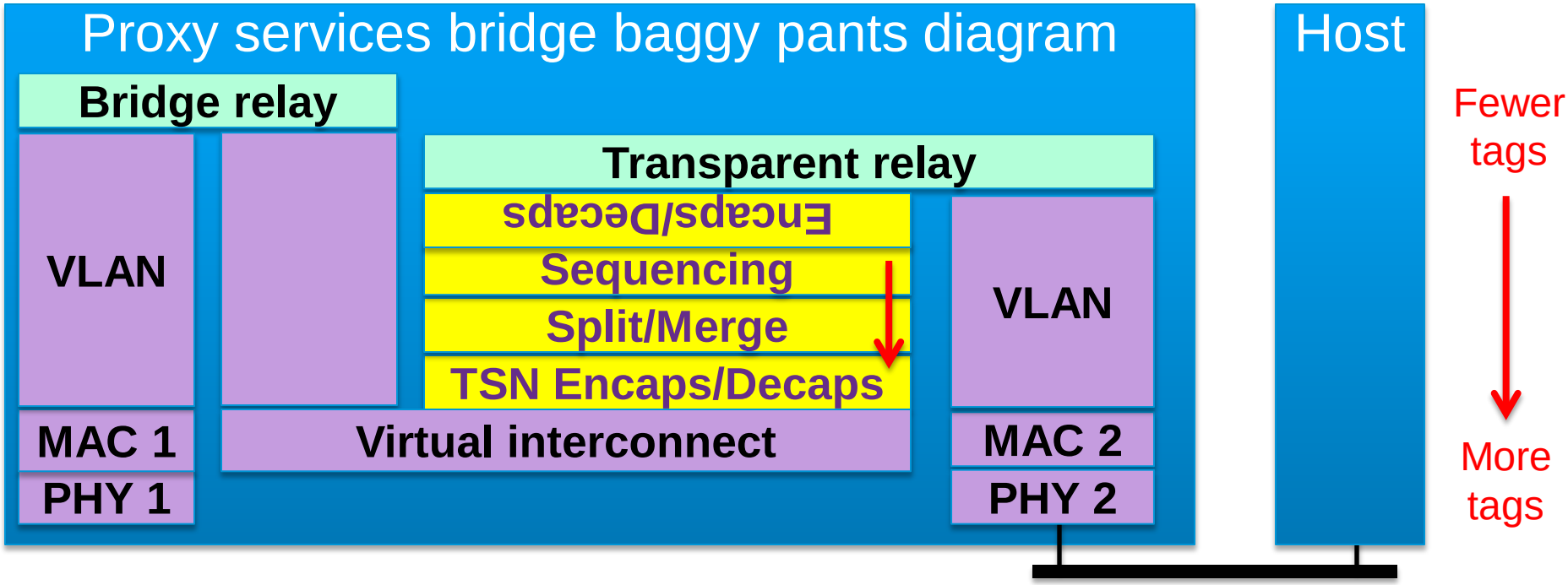


Extra “free form” Encaps/Decaps layer

- This extra Encapsulation / Decapsulation layer is responsible for identifying the flows coming from the host that are to be placed into TSN circuits.
- If all traffic is a single TSN circuit, this is a no-op. If IP 5-tuple inspection is required, then no actual encapsulation / decapsulation is required, just packet inspection.

Proxy bridge stack

- As we know in IEEE 802.1, we can turn the functions “right side up” by attaching a relay shim that parallels the host stack

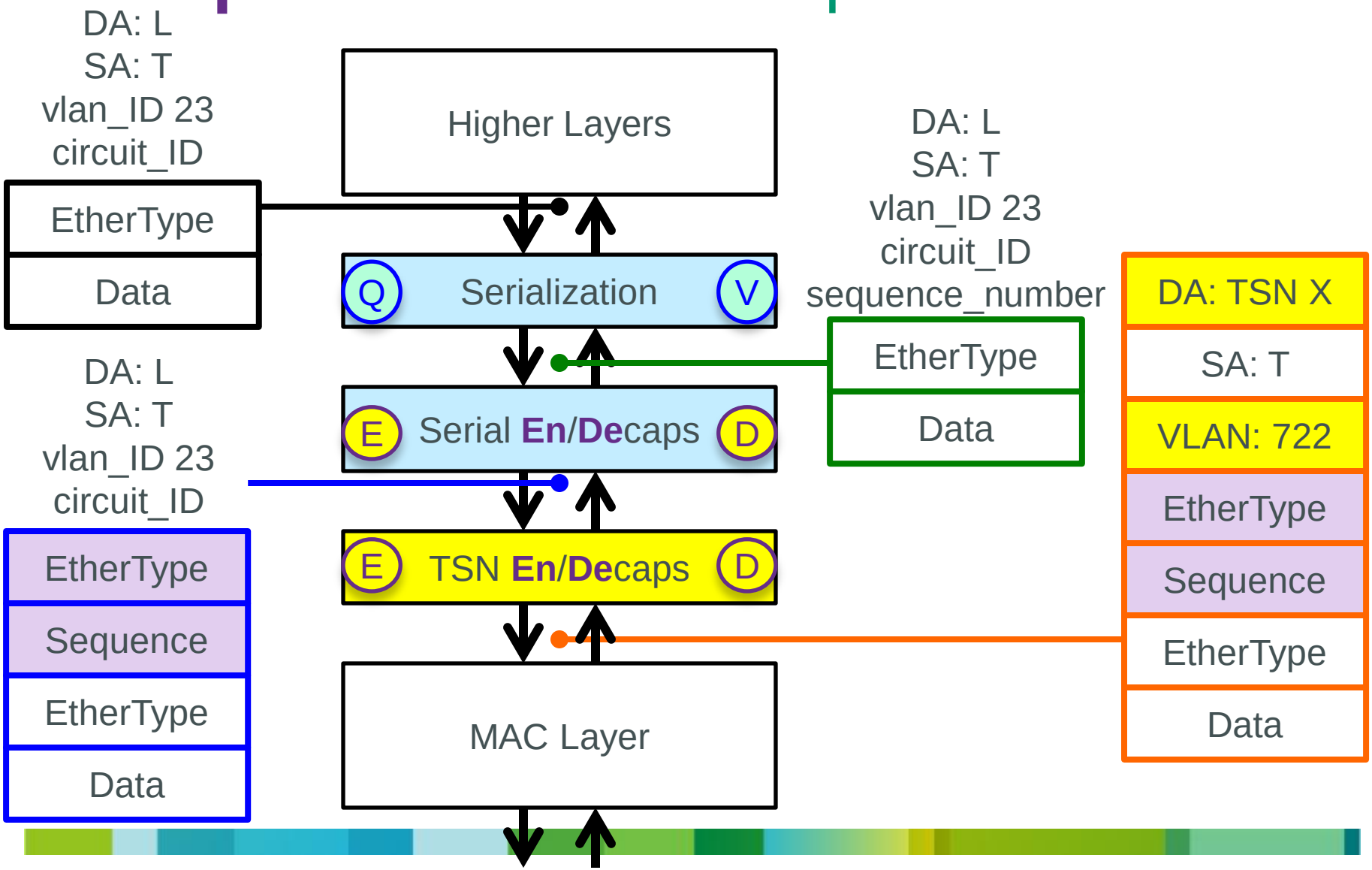


Alternative TSN Encapsulations

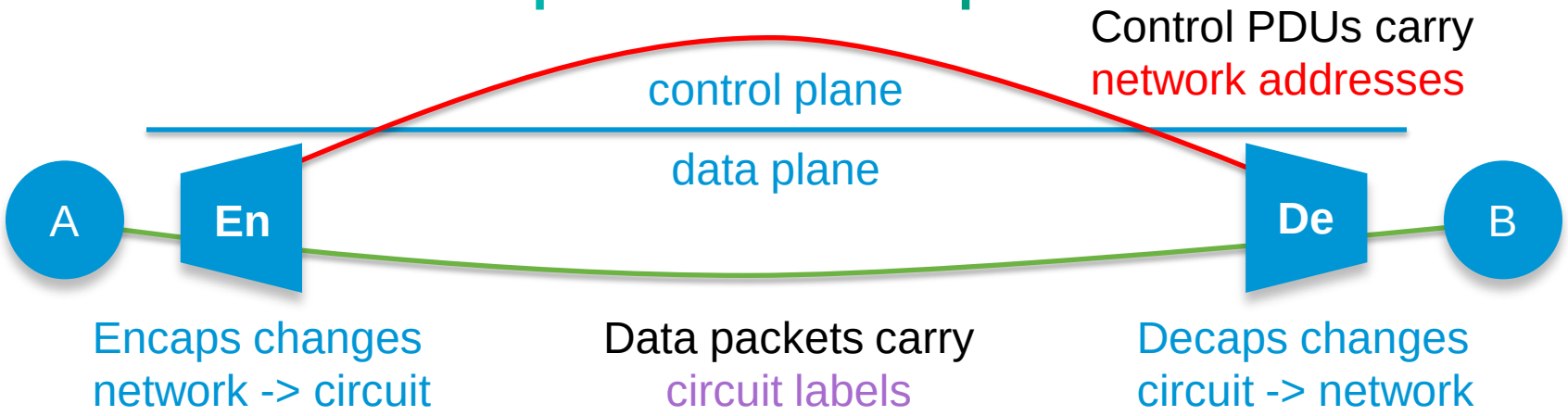
1. **Sequenced TSN** encapsulation

- We have a protocol for Circuit Identification for the simple Ethernet end-to-end case, the current TSN encapsulation
- We have supplied no protocol for the Sequencing function, yet.
- This could be simply a tag with a sequence number.

1. Sequenced TSN encapsulation

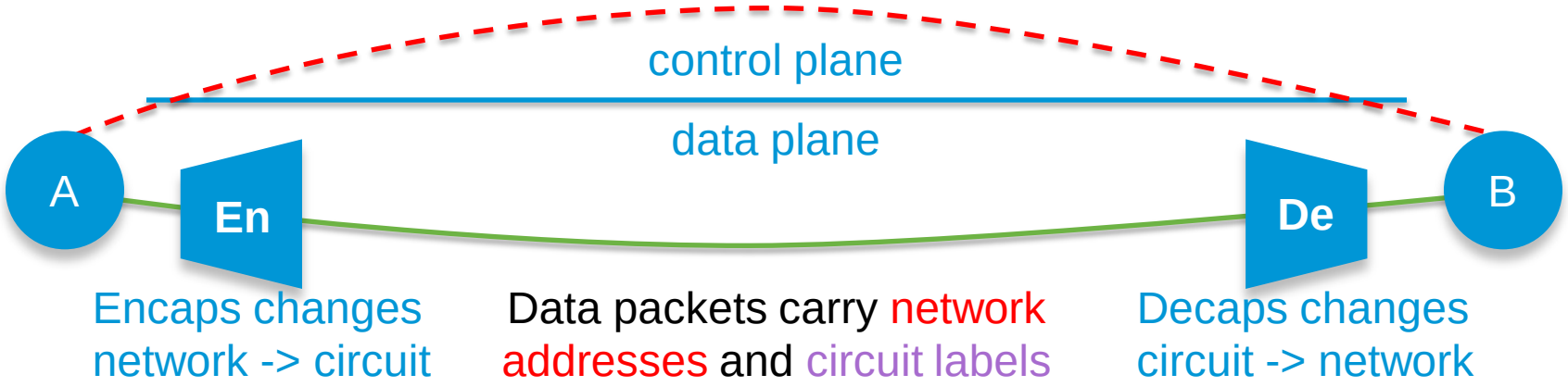


Are other encapsulations possible? Yes!



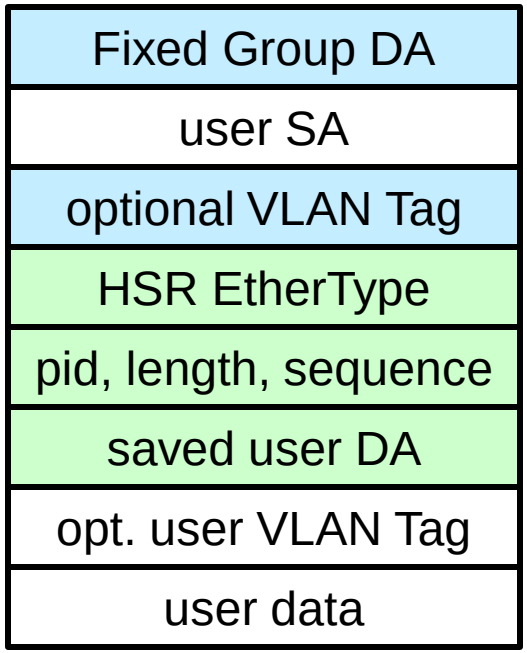
- To use the current AVB format, while making TSN services transparent to the user, the Encapsulation function destroys information, and the Decapsulation function restores it.
- Let's call this **out-of-band tunneling**.

Are other encapsulations possible? Yes!



- We could do **in-band tunneling**, and encapsulate the B's address and VLAN in the data frame, itself.
- There are existing, applicable protocols that work both ways.

2. HSR seamless redundancy

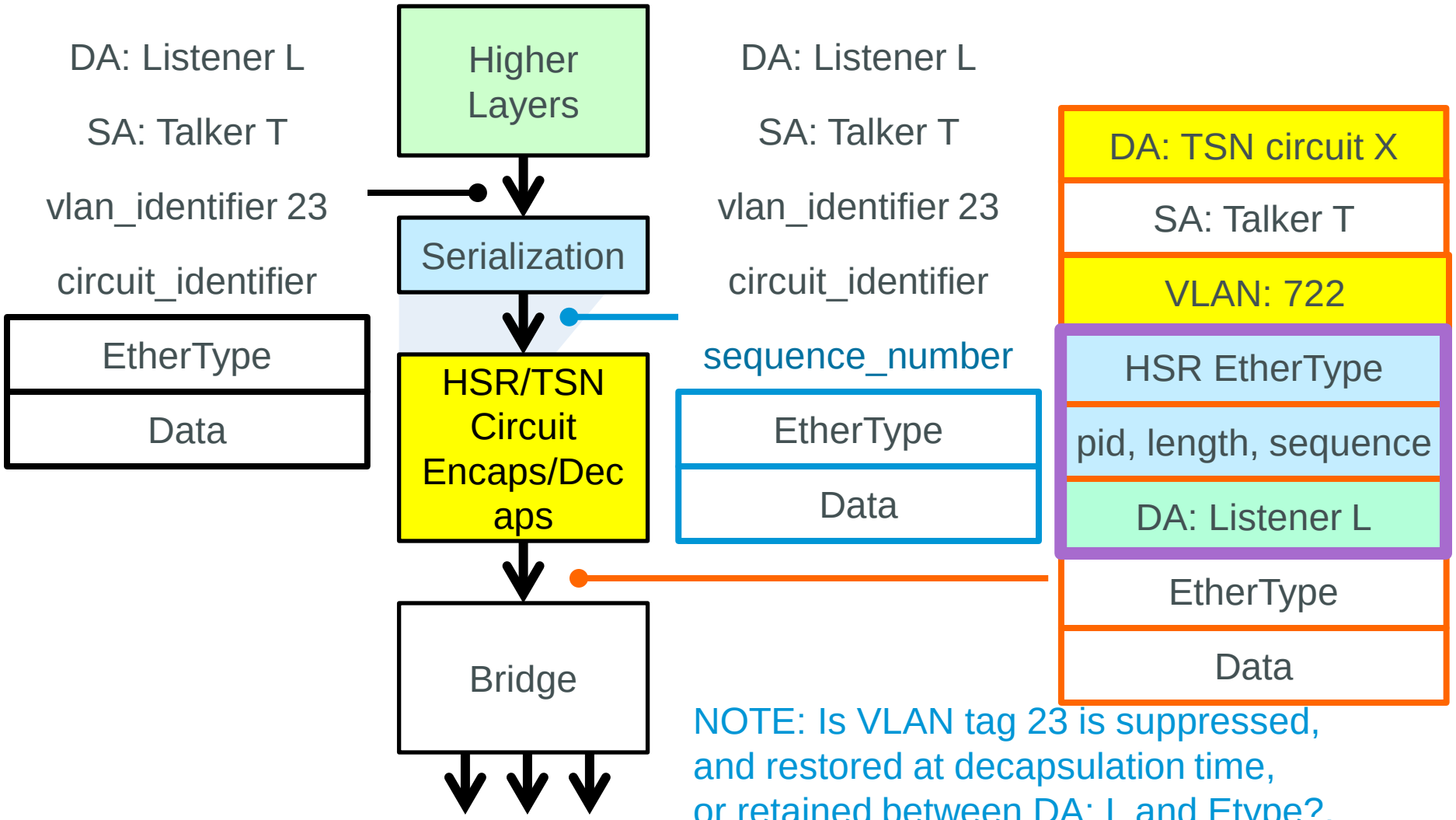


- The **IEC 62439-3 High-Speed Seamless Redundancy (HSR)** encapsulation provides in-band tunneling.
- Almost. On the good side:
  - HSR encapsulates the original destination MAC and VLAN, rather than using the data plane.
  - HSR includes a sequence number for seamless redundancy. (That **is** the name of the protocol!)

2. HSR**-like** seamless redundancy

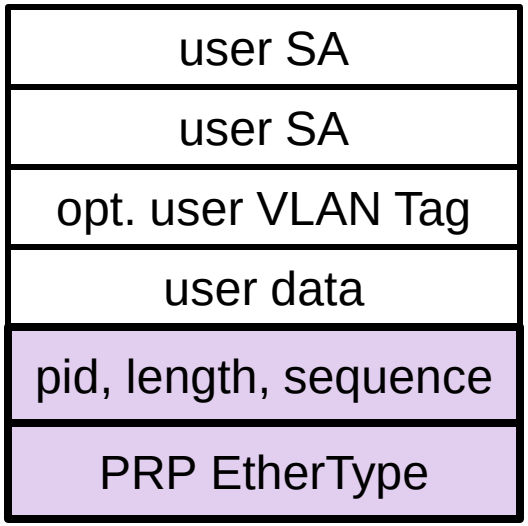
- But! HSR only works if the network is an HSR ring, because the TSN circuit address has been buried behind the fixed HSR DA.
- To make HSR work over 802.1 networks:
  - Change to the PDU format:
    - We substitute the TSN circuit DA for the HSR fixed DA.
  - Changes to the use of that format:
    - The Link Redundancy layer does not forward packets – that is left to a bridge function, below it.
    - Don’t encapsulate VLAN tags

2. HSR-like seamless redundancy



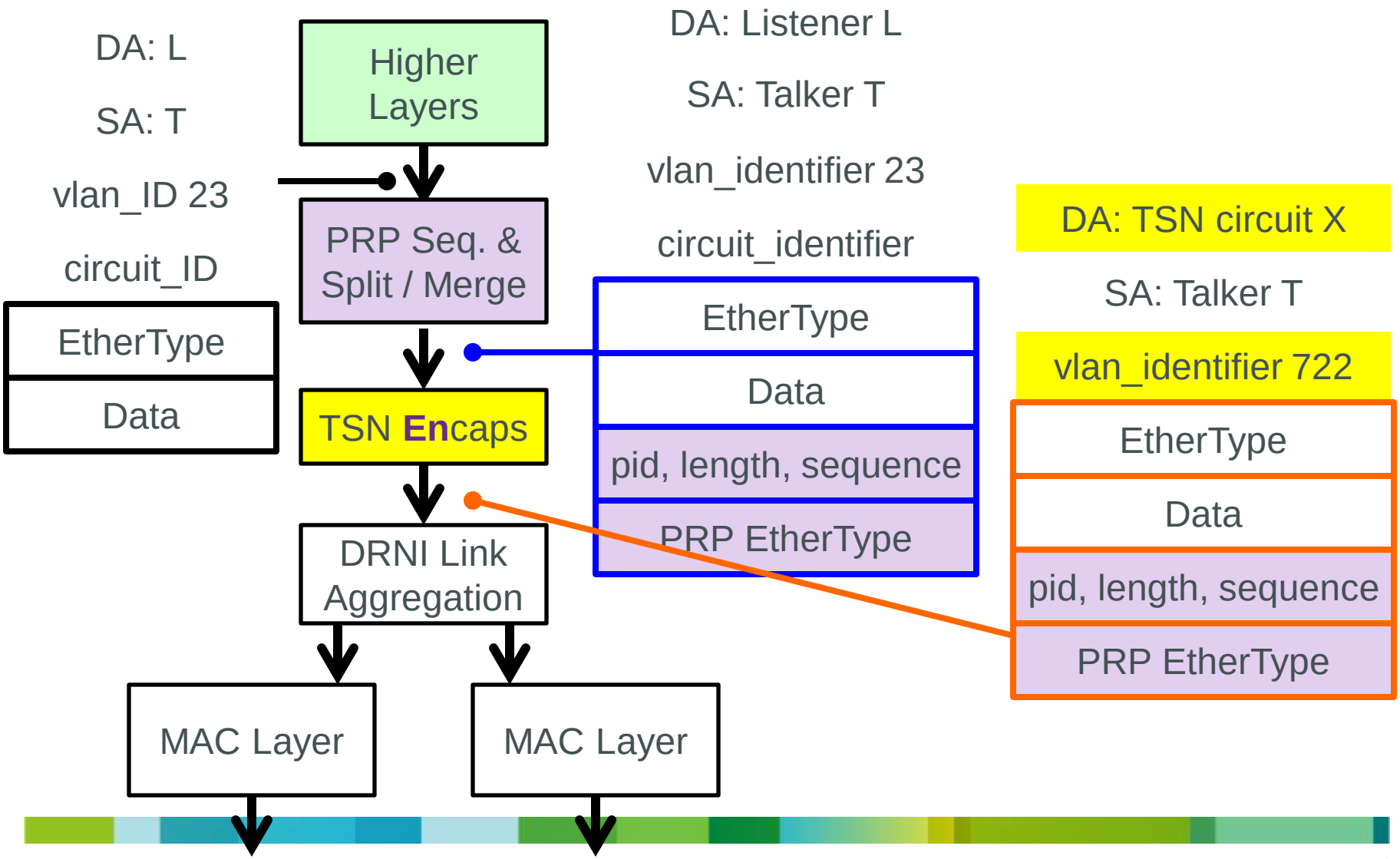
NOTE: Is VLAN tag 23 is suppressed, and restored at decapsulation time, or retained between DA: L and Etype?.

3. PRP seamless redundancy + TSN



- The **IEC 62439-3 Parallel Redundancy Redundancy (PRP)** encapsulation will work with the TSN encapsulation.
- There are serious issues with an 802.3 frame trailer. Let's leave that for another discussion.
- TSN supplies the circuit ID, and PRP supplies the split/merge capability.

3. IEC 62439-3 PRP + TSN



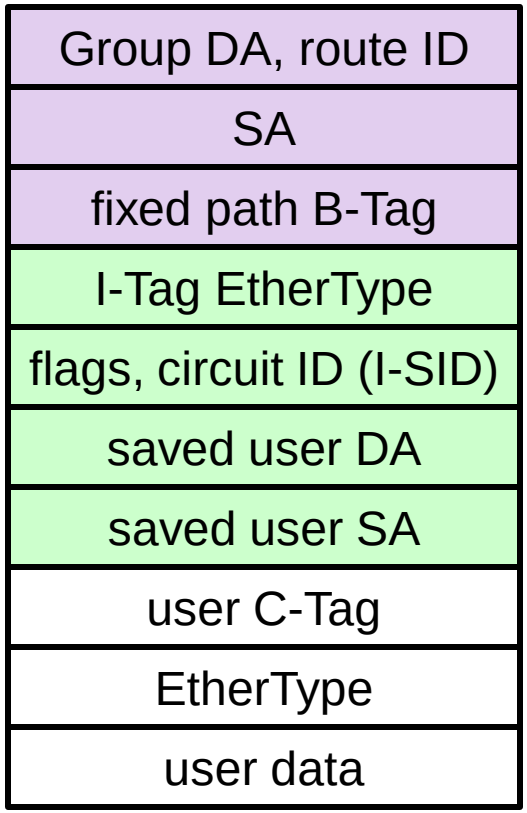
3. IEC 62439-3 PRP + TSN

- This formulation decomposes the “PRP Link Redundancy” sublayer defined in IEC 62439-3 into a “PRP Sequencing and Split/Merge” sublayer, a TSN encapsulation sublayer, and a DRNI sublayer.
- The reader may also notice that, when a trailer is used, the ordering of the layers becomes ambiguous.

3. IEC 62439-3 PRP + TSN

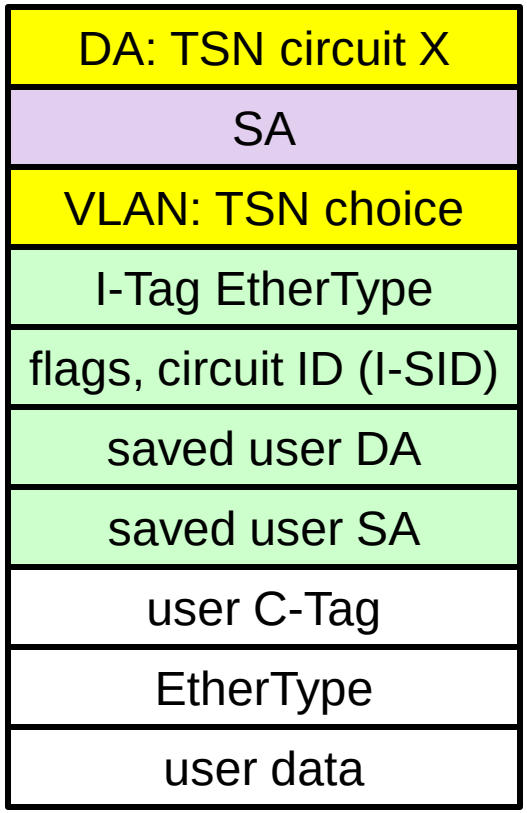
- PRP was intended to work only over separate networks; one port is connected to one network, and the other to another network. But, we can make it work over a single 802.1 network.
- It works, because the TSN circuits provide the logical equivalent of separate networks.

4. PBB-TE for TSN



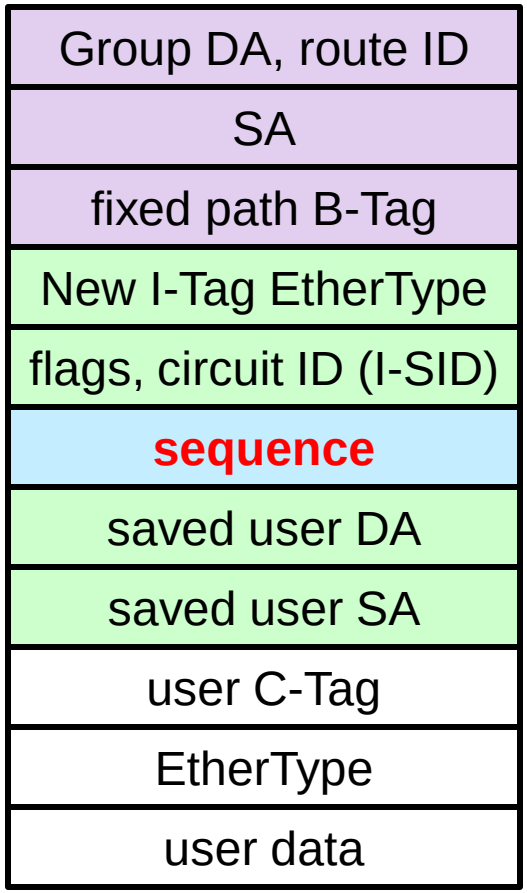
- MAC-in-MAC (802.1ah) solves the circuit ID and fixed path problems using in-band tunneling.
  - It encapsulates the data now carried in the control plane.
  - But, it has no sequence number for seamless redundancy.

4. PBB-TE for TSN



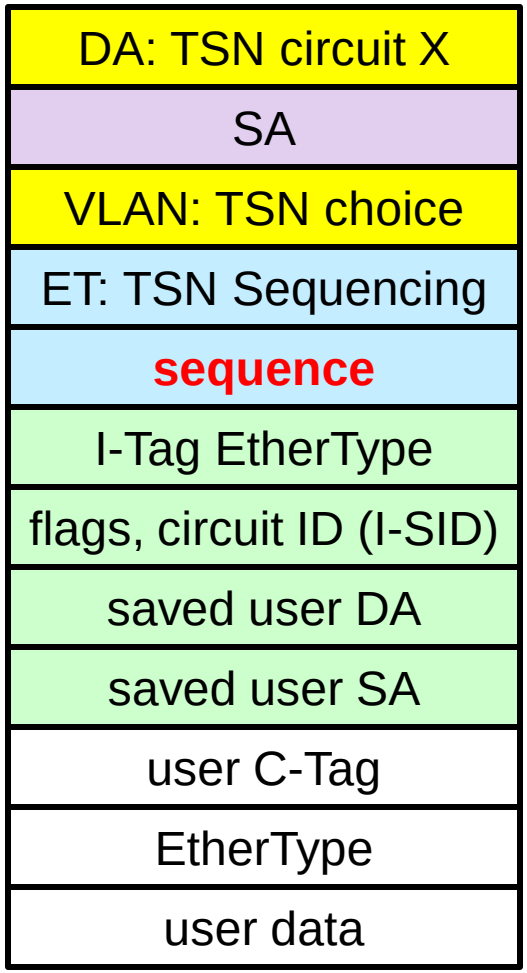
- MAC-in-MAC (802.1ah) solves the circuit ID and fixed path problems using in-band tunneling.
  - Of course, we would use a TSN address and VLAN, where appropriate.

4. PBB-TE for seamless redundancy (1)



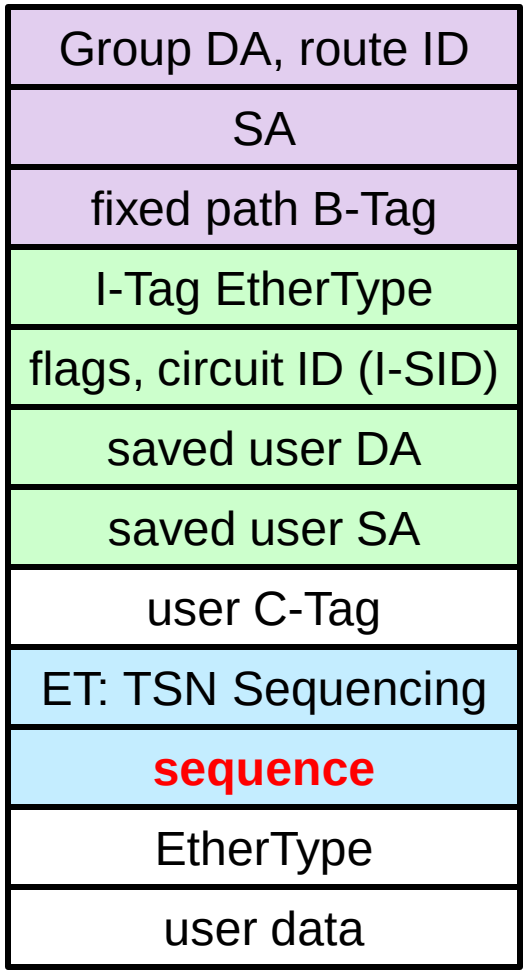
- We could put the sequence number in a new I-Tag format.
- Note that the sequence number uses the I-SID for the circuit ID, so the normal PBB-TE outer addresses can identify the circuit to the backbone bridges..

4. PBB-TE for seamless redundancy (2)



- We could add an additional tag for the sequence number.
- Now, the DA/VLAN are the circuit ID, just like for the Sequenced TSN encaps.
  - (This is nitpicking; the PBB-TE {DA, VLAN} is a circuit ID.)
- The sequence tagging choice makes a big difference in the position of the Sequencing sublayer in the CBP stack.

4. PBB-TE for seamless redundancy (3)

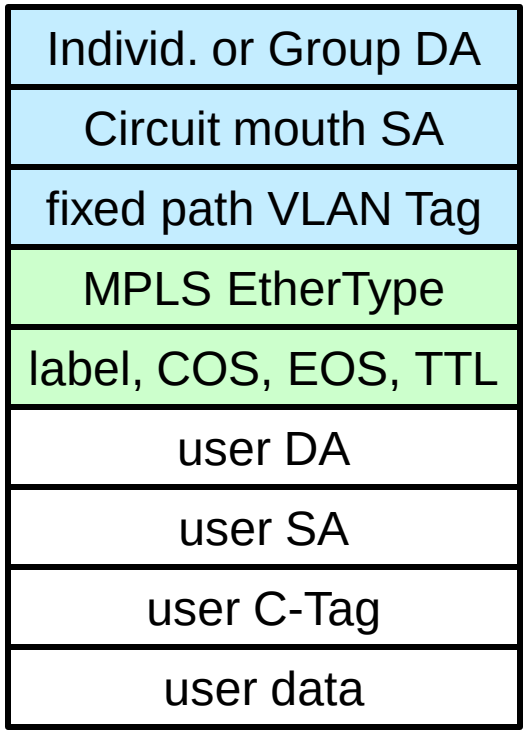


- We could add a tag for the sequence number in the customer frame.
- Now, the whole MAC-in-MAC is the circuit encapsulation.
- (This is the option that seems to make the most sense to this author.)

4. PBB-TE for seamless redundancy

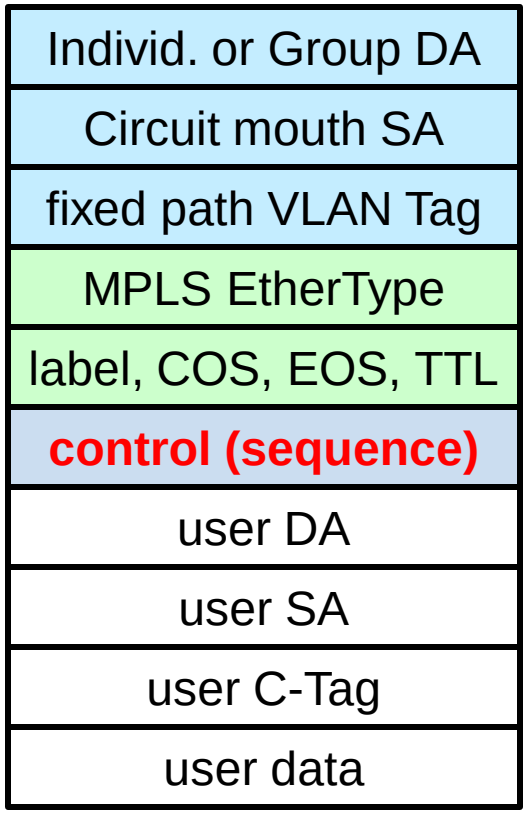
- Placing the sequence number is outside the I-tag is great if you do all of the seamless redundancy in the edge bridges, or if the host systems include the Customer Backbone Port (CBP) and do their own MAC-in-MAC.
- Making the sequence number part of the I-tag has the same consequences for the host.
- Putting the sequence number down in the customer frame simplifies TSN-aware hosts, but makes the sequence number invisible to the backbone. PBB-TE is then purely a circuit labeling technology.
- Of course, backbone/customer interworking functions, similar to IEEE Std 802.1Qau, are possible.

5. MPLS Ethernet Pseudowire



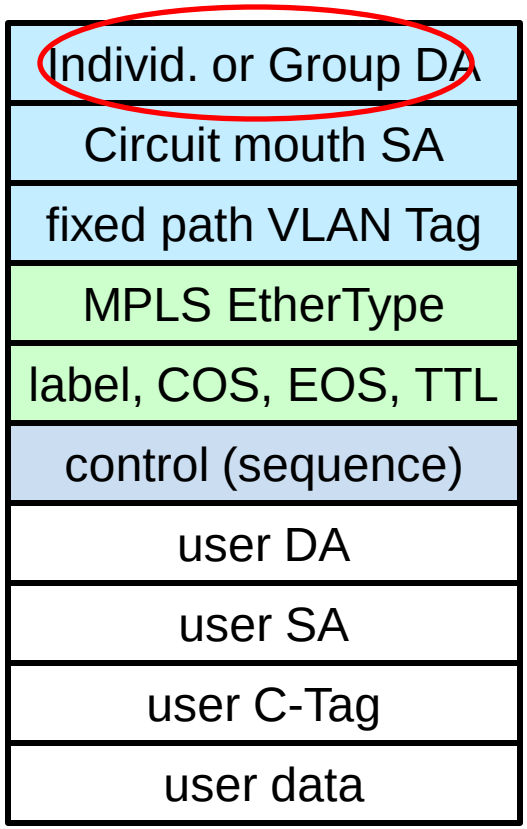
- An **MPLS pseudowire** provides almost exactly the same encapsulation structure as 802.1ah.
- Like the current AVB encapsulation, it provides out-of-band tunneling.

5. Pseudowires for seamless redundancy



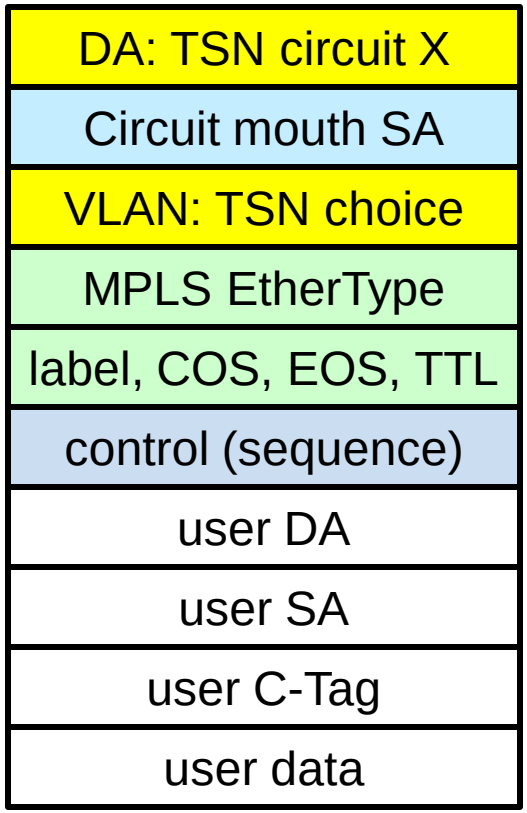
- Plus, Pseudowires have an optional control word that provides a **sequence number for seamless redundancy**.

5. MPLS Ethernet Pseudowire



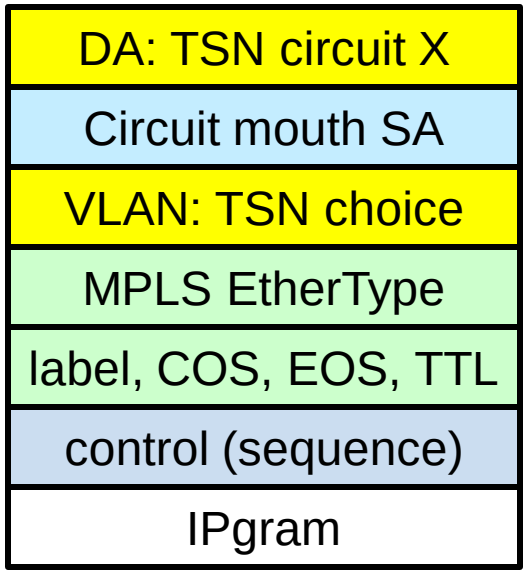
- There is one **problem**: MPLS uses either the unicast next-hop destination, or a fixed Group DA.

5. MPLS Ethernet Pseudowire



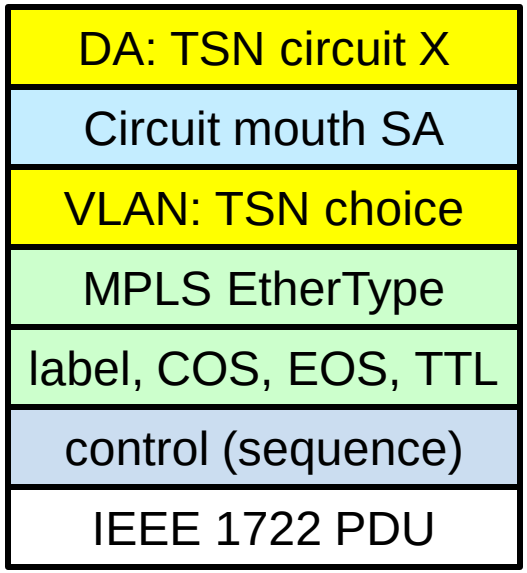
- But, we just talked about the TSN Encaps/Decaps layer changing the outer MAC address!
- So, this is the encapsulation we would use.

6. MPLS IPgram Pseudowire



- Although Ethernet pseudowires are common, they are not the only kind.
- A pseudowire format exists for a bare IPgram, also.
- **This is close to the size of the L2-only encapsulation, and works for bridged, routed, or mixed networks.**

7. IEEE 1722 Pseudowire



- All it takes is a code point from IANA to create an IEEE 1722 pseudowire.
- This is both an L2 and L3 format. If used between TSN-aware endpoints, the **IP/UDP header** (for end-to-end addressing) **could be omitted** from the actual frame, just as the MAC addresses are omitted from an L2 end-to-end TSN frame.
- There is no 1722 EtherType, either.

8. Ordinary multicast

- **Any multicast stream**, whether an L2 multicast or an IP multicast stream, **can be a TSN circuit**.
- If it is on a pinned-down path, it must be on a TSN VLAN, not a normal 802.1Q topology-controlled VLAN. (This can be handled by a TSN Encapsulation layer that changes only the VID.)
- Whatever VLAN it is on, it must have a unique Destination MAC address. The address does not have to come from any particular space – the IP multicast space is fine, as long as there are not multiple streams with the same MAC DA.

9. No protocol at all

- OpenFlow can recognize a circuit based on common fields, such as the MAC addresses or IP 5-tuple.
- Any number of bridges have “Access Control Lists” (ACLs) that can inspect a frame and take special action.
- So, one can always leave the original frame intact, and use frame inspection to identify the circuit so that special actions can be taken.
- **But, this layering model is still important, you’re still doing circuits, and P802.1Qcc still has to change.**

Issue with tunneling VLANs

- Bridges can change the VLAN ID as a frame traverses the Bridged LAN. At the very least, Talkers and Listeners may or may not use VLAN tags.
- Therefore, **any protocol such as HSR or PBB-TE that encapsulates a C-tag has a problem, if the underlying network does not use the same encapsulation**. Either:
  - The TSN Encaps encapsulates the right VID (impossible for a multicast); or
  - The TSN Decaps fixes up the encapsulated VID, based on control plane information.
- Of course, one can always emulate a separate point-to-point Ethernet port, with a separate IP stack, at each end of a PBB-TE connection.

Summary

Summary 1/2

- When you realize that TSN is all about encapsulating circuits, everything just works.
  - No need for deep packet inspection. No problems with existing IP stacks, too many VLAN IDs, MAC address learning, fixed paths, or backwards compatibility.
- We must align P802.1Qcc with circuit encapsulation.
- We should define the Sequencing, TSN Encaps/Decaps, and Split/Merge sublayers.

Summary 2/2

- Existing protocols support TSN circuits.
  - TSN, HSR, MPLS, PBB-TE, ordinary multicasts, no protocol at all, and many more …
- Existing protocols have sequence numbers that support seamless redundancy.
  - HSR, PRP + TSN, Ethernet pseudowires, IPgram pseudowires, TCP, IEEE 1722, more ...
- We can invent a new Sequencing tag protocol only for L2 end-to-end seamless redundancy.
- End-to-end L2 encapsulations work if they maintain the “natural” L2 relationships, or if they simulate long-range point-to-point “wires”.

Thank you.