Layering for the TSN Layer 2 Data Plane

Norman Finn
Version 3

Feb. 17, 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

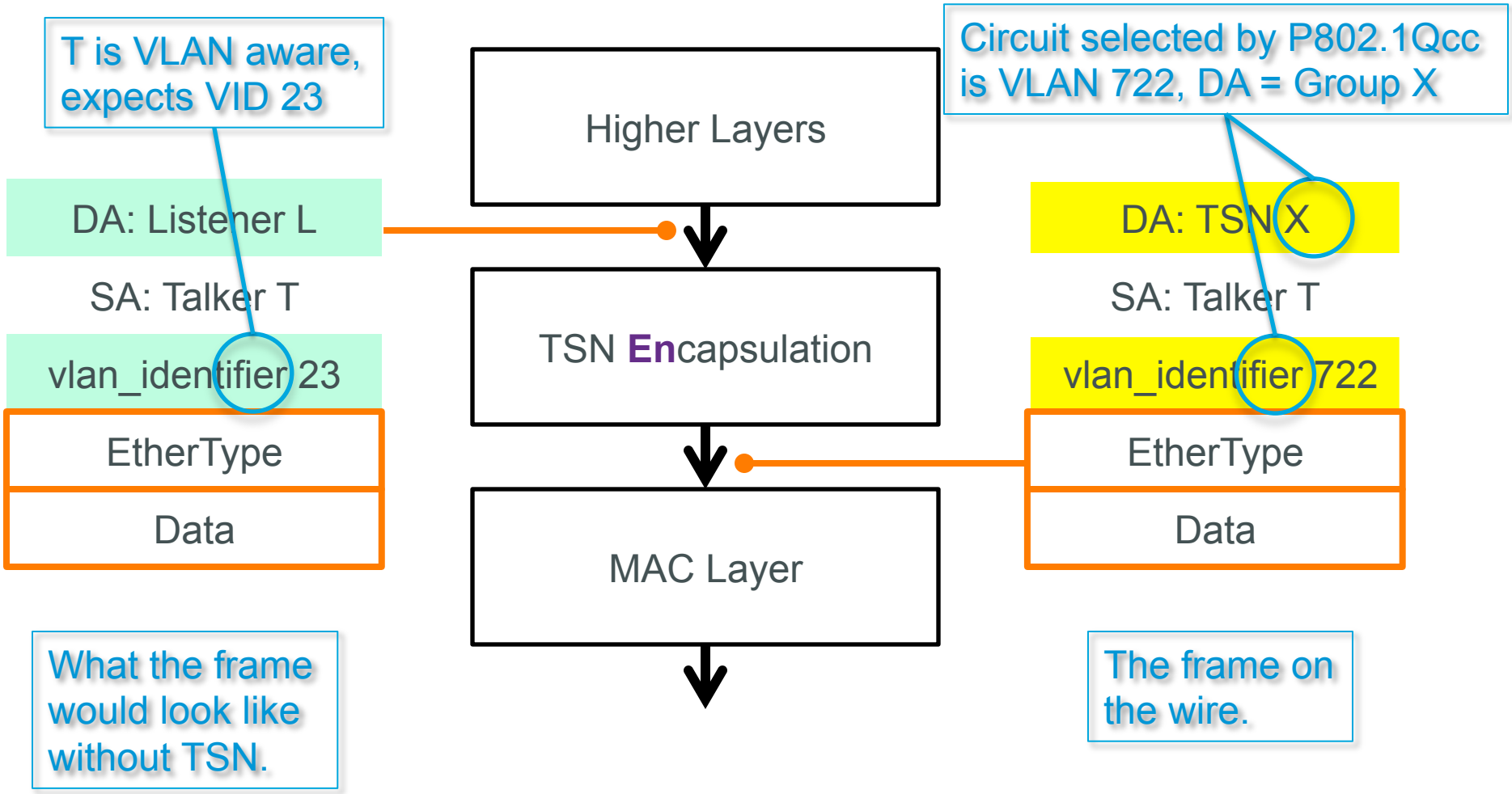
- This is tsn-nfinn-L2-Data-Plane-0214-v03. It offers an improved model for layering the AVB/TSN concepts, which leads to a set of choices for answering the data plane questions A, B, and C. Contents of this presentation:
  - What's broken about AVB/TSN?
  - What's the Fix?
  - Sublayer structure for TSN
  - Alternative TSN Encapsulations
  - Summary
- This presentation does not make choices, and so cannot advance to Question D, protocols.

NOTE

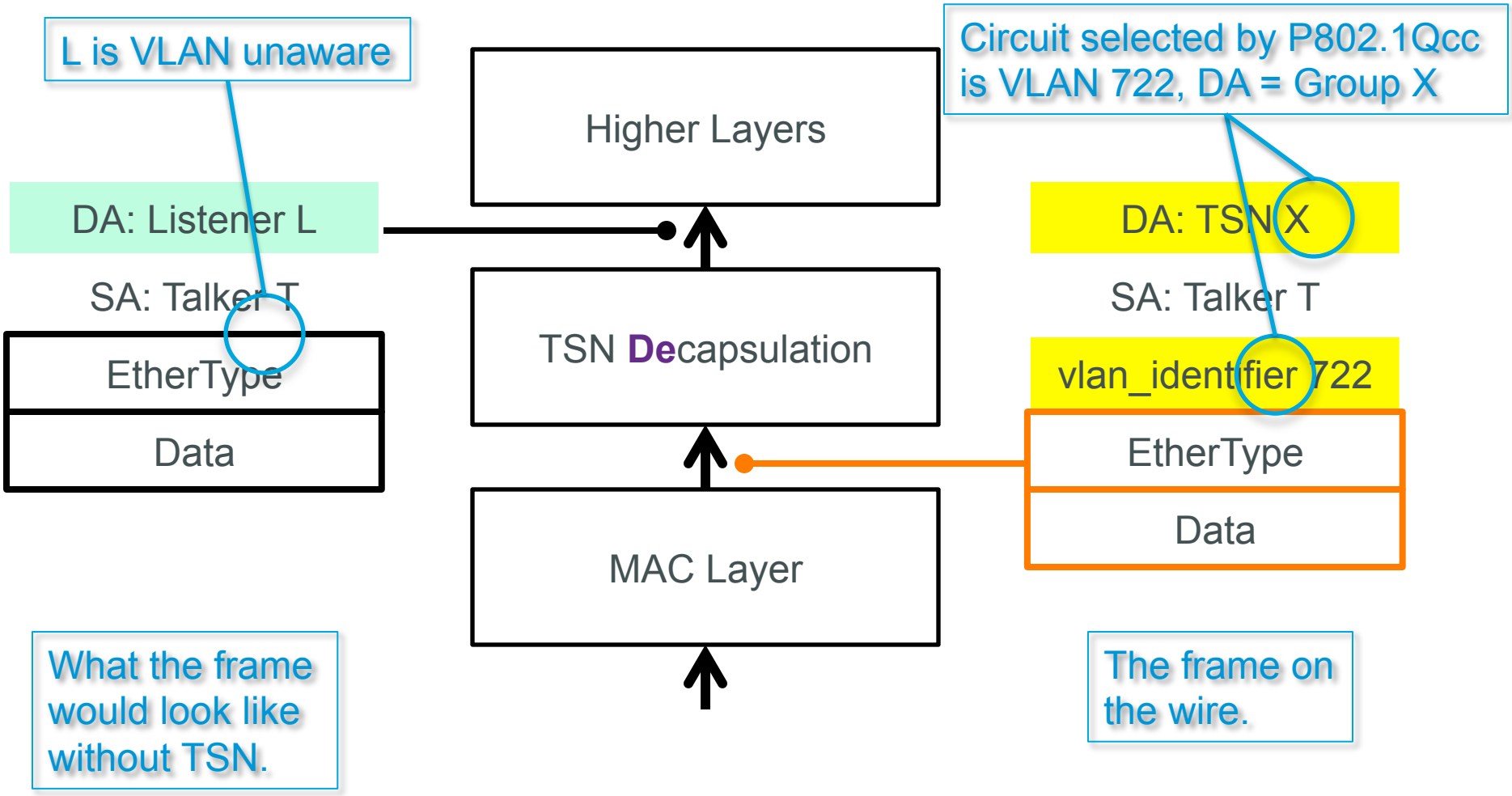
- The previous version of this presentation, tsn-nfinn-L2-Data-Plane-0114-v02, had the order of the TSN Encaps/Decaps and Serialization layers reversed.
- This was discovered by the author when the details of how the layers work was expanded for this version of the presentation.
- It was an easy mistake to make, because the outer MAC addresses are outside the hierarchy of tags.

TSN Sublayers

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



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



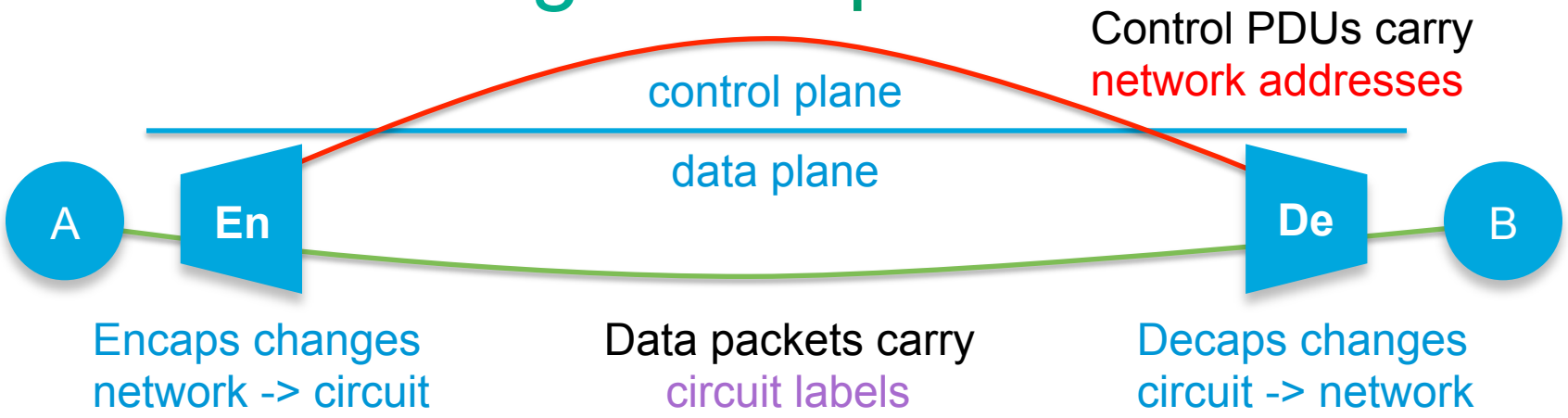
How does this get setup?

- **A**'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?

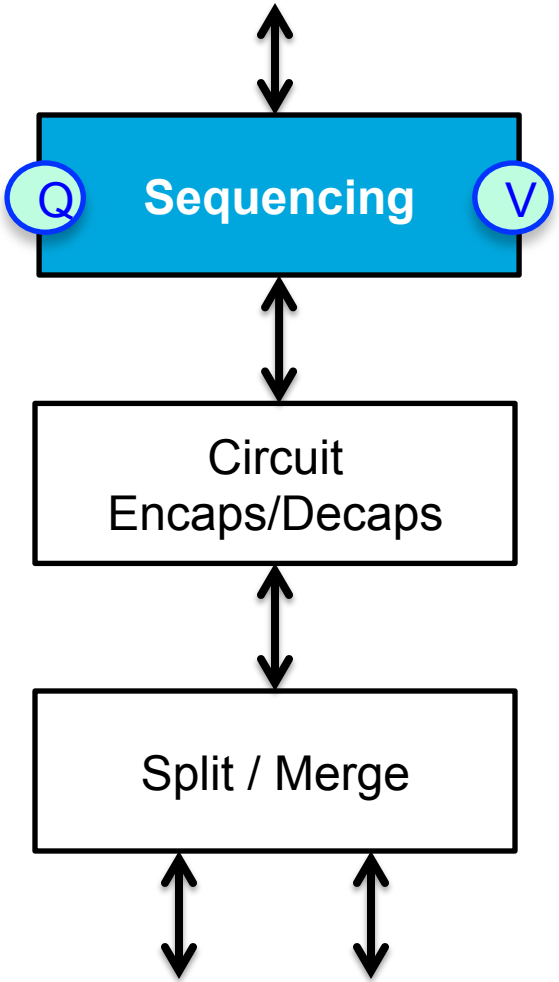
1. Revise 802.1Q to have the TSN Encaps/ Decaps layer, working as described.
2. Fix P802.1Qcc to:
   - Take the connectionless target {VID, MAC address} pair as an input.
   - Modify that pair, as needed, as P802.1Qcc proceeds through the network.
   - Give the circuit {VID, MAC address} to the endpoints,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

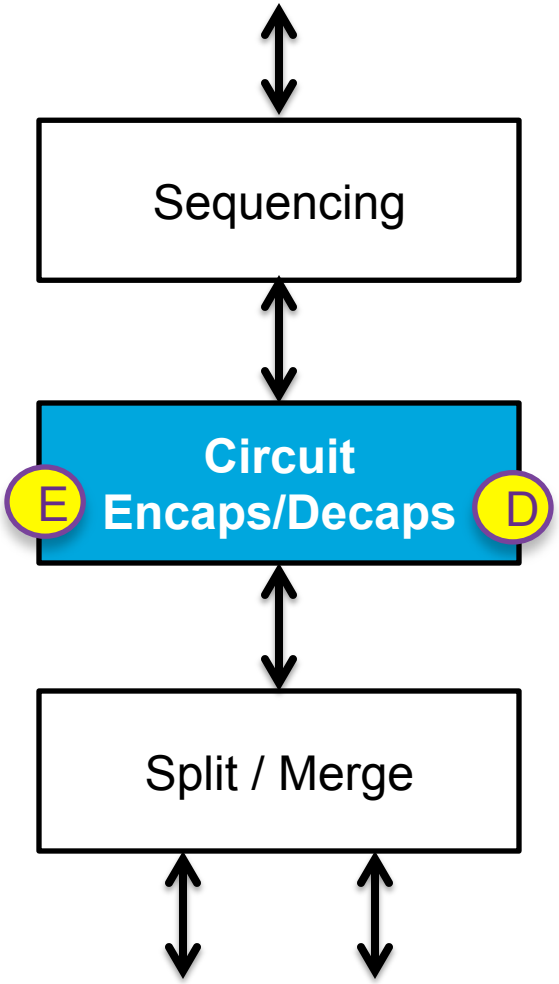
Layering

Functional elements required for TSN



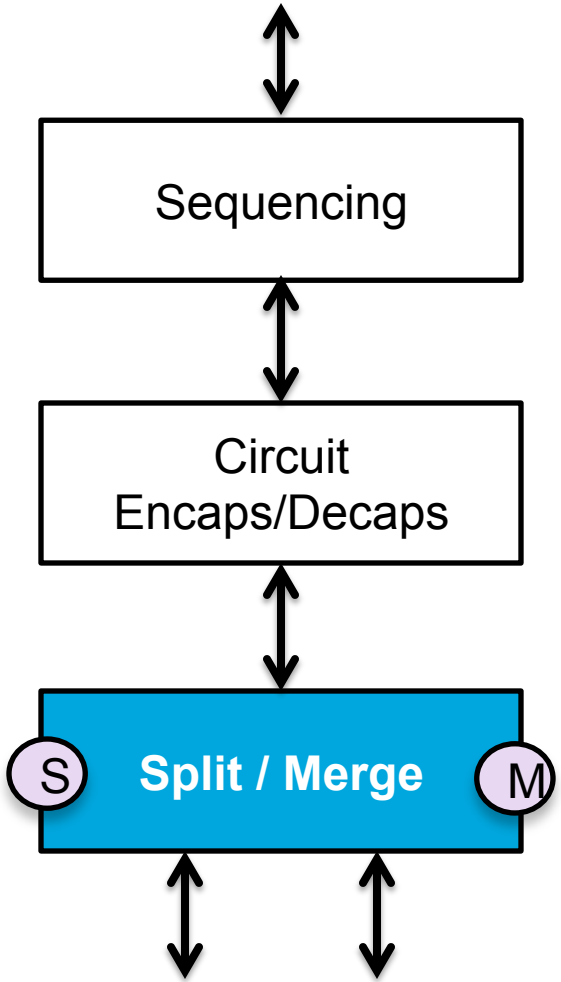
- **If** a circuit requires either seamless redundancy or long-range out-of-order delivery protection, then:
  - Packets are sequenced on transmit. (Q)
  - Duplicates are discarded on receive; out-of-order may be discarded. (V)
  - Packets may be buffered and reordered on receive.

Functional elements required for TSN



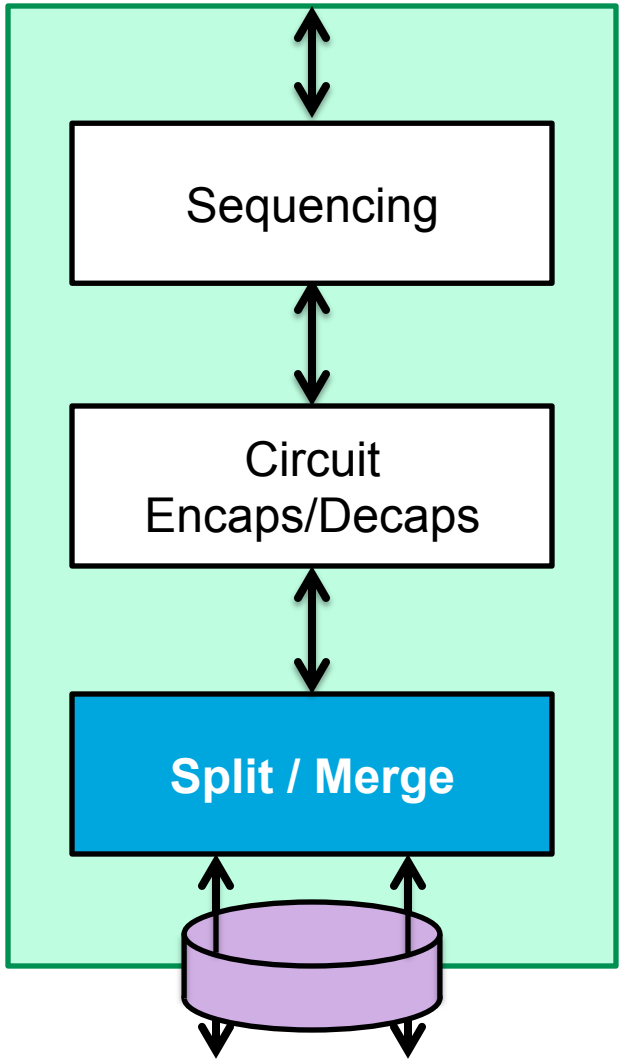
- Individual circuits must be identified, and packets encapsulated (E) and decapsulated (D), for:
  - Fixed paths;
  - Per-circuit resources;
  - Seamless redundancy.

Functional elements required for TSN



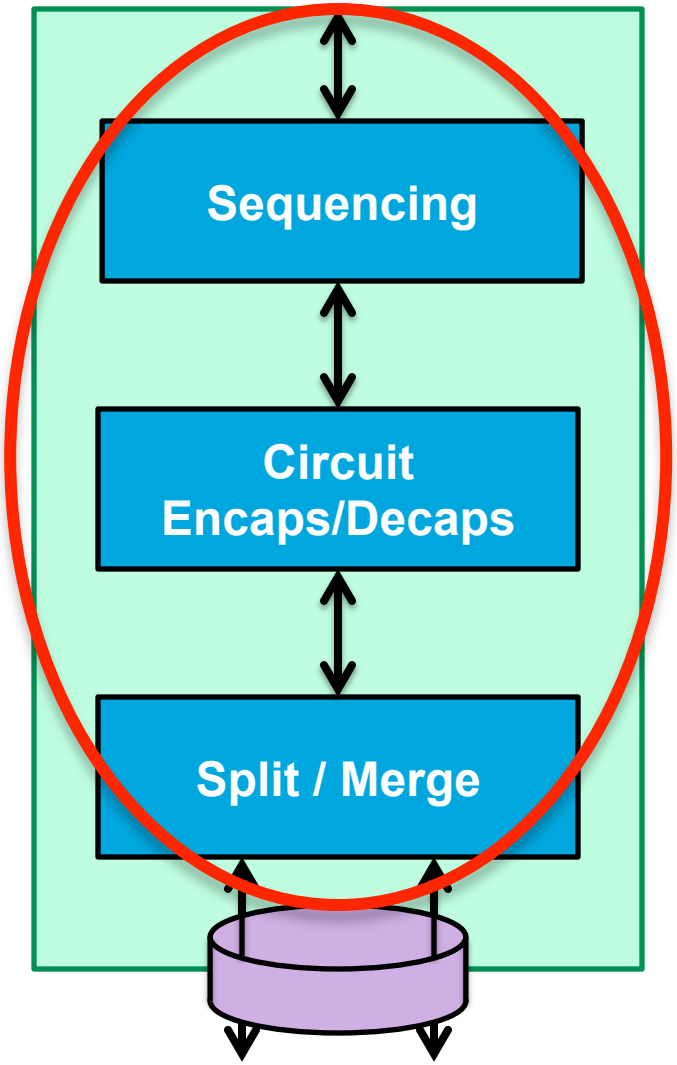
- **If** circuit requires multiple paths, then its packets must be:
  - Replicated, (S) and perhaps relabeled, for transmit to one or more ports.
  - Merged into one port, (M) and perhaps relabeled, (but **duplicates are not discarded**) on receive.

Functional elements required for TSN



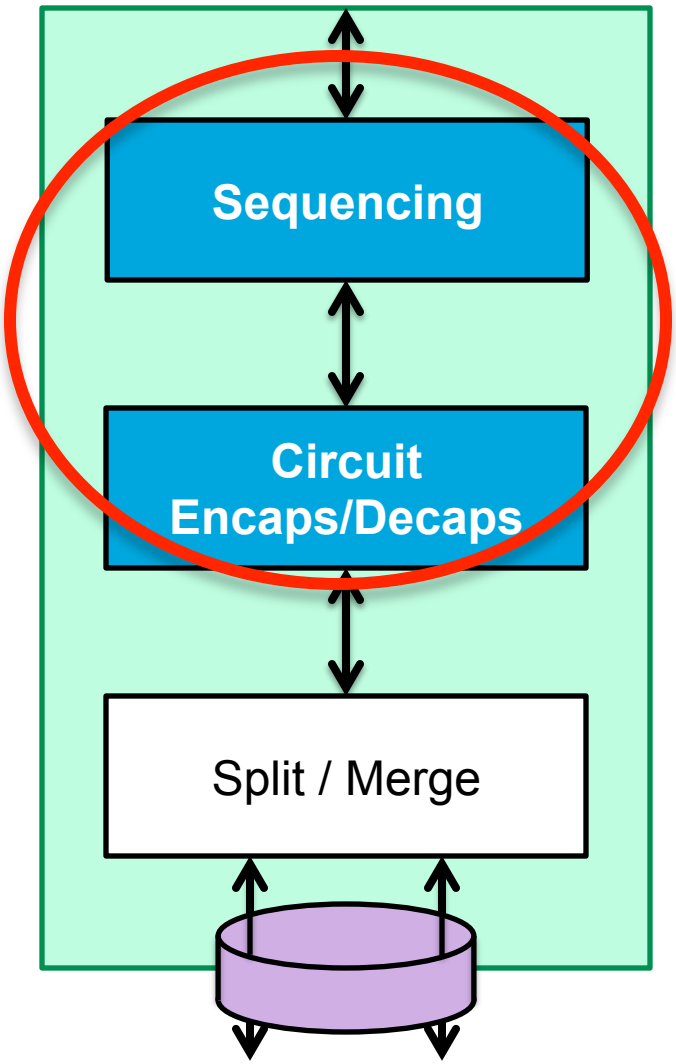
- Note that the Split Merge function does not require two physical ports.
- It may replicate / merge the circuits by using different explicit in-band markers, leaving it to normal networking to make the physical split.

Functional elements required for TSN



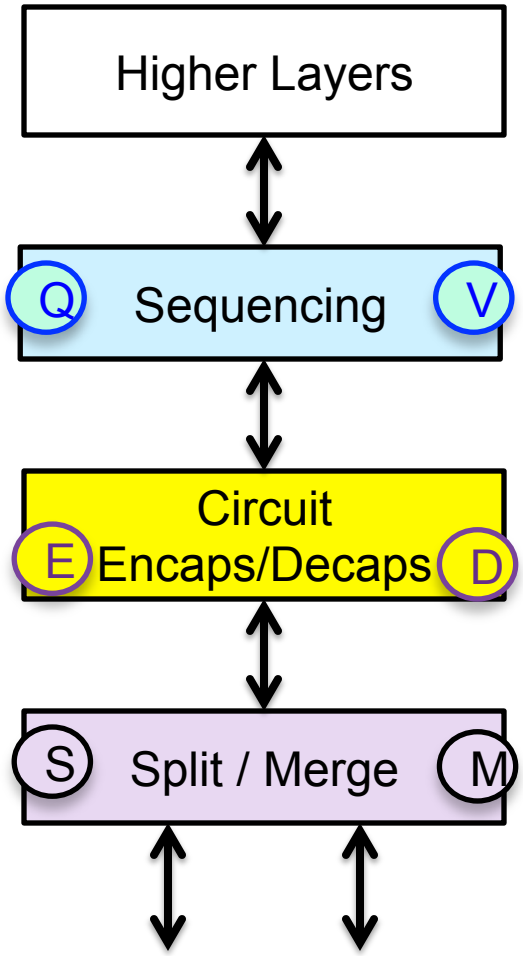
- These layers may be combined to provide a **seamless redundancy** feature.

Functional elements required for TSN



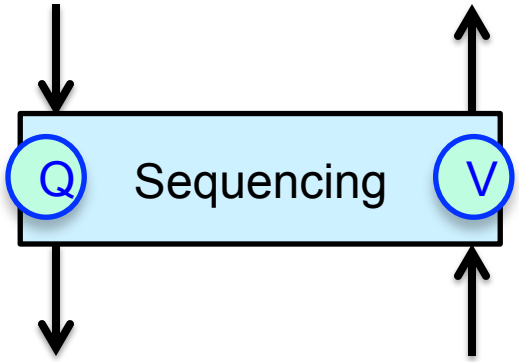
- Some encapsulations (described later in this deck) combine the Circuit encapsulation with the sequencing.
- This leaves the Split / Merge functions to do packet redirecting and/or relabeling.

Functional elements required for TSN



- The ordering is important.
- We will see why this ordering works as we go through the details.

Sequencing sublayer

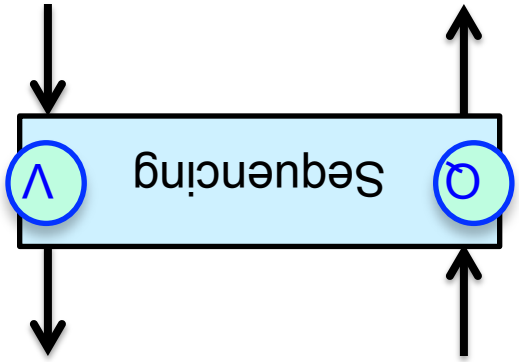


Addresses, Data, circuit_identifier, sequence_number

Addresses, Data*, circuit_identifier

- The Squencing sublayer inserts (Q, on the way down) a sequence number into the Data, and removes it (V, on the way up).
- If no sequence number supplied going down, the sequencing sublayer supplies one.
- Typically, it needs the circuit_identifier interface, because the sequence_number is per-circuit.

Sequencing sublayer

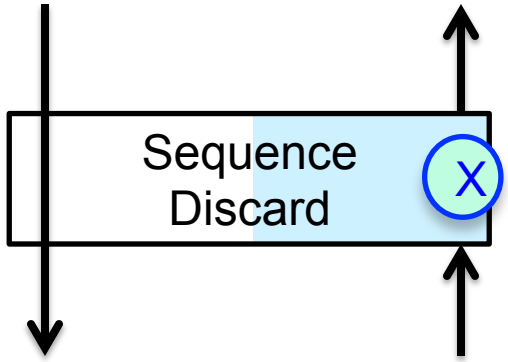


Addresses, Data, circuit_identifier

Addresses, Data*, circuit_identifier, sequence_number

- As was the case for the Circuit ID sublayer, we sometimes have to turn the Sequencing layer upside down in our diagrams.

Sequence Discard sublayer

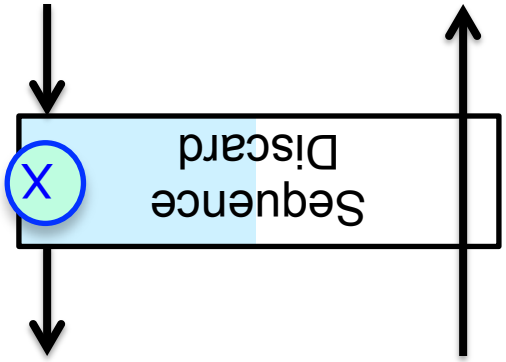


Addresses, Data, circuit_identifier, sequence_number

Addresses, Data, circuit_identifier, sequence_number

- The Squence Discard sublayer (X, 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.
- This shim discards out-of-order or duplicate packets, and may buffer and reorder them.

Sequence Discard sublayer

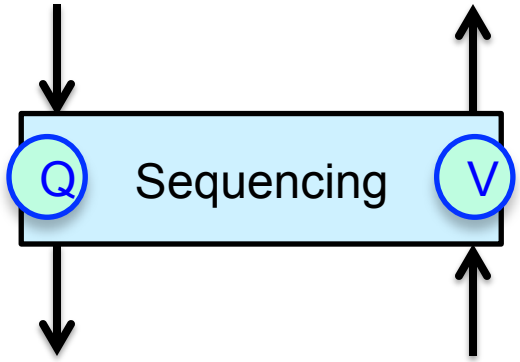


Addresses, Data, circuit_identifier, sequence_number

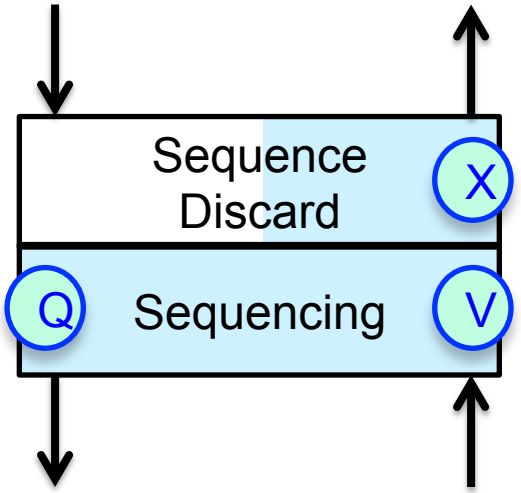
Addresses, Data, circuit_identifier, sequence_number

- The Sequence Discard sublayer can operate in the other direction, of course. Its direction of operation will be clear from the context.

Sequence Discard sublayer

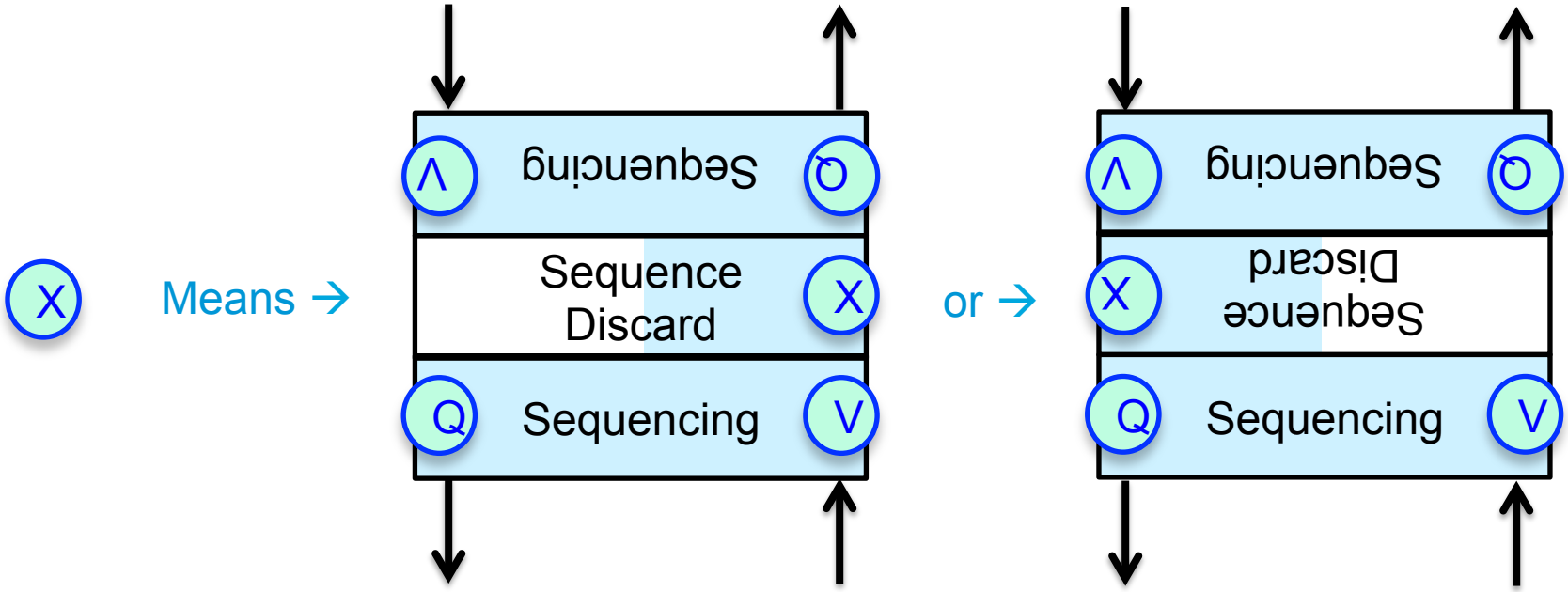


Usually means →



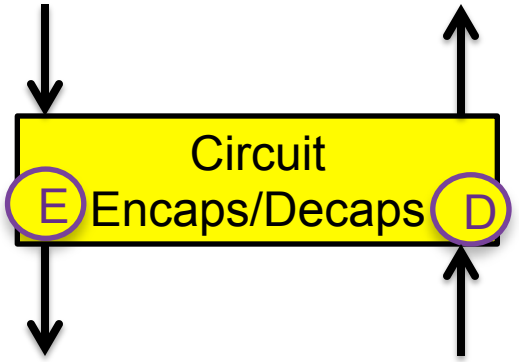
- To prevent clutter, unless otherwise noted, the sequencing removal action (V) will usually imply the Sequence Discard sublayer (X) in this deck.

Sequence Discard sublayer



- Similarly, a bare Sequence Discard function (X) is really a "Peek at the sequence number and discard" function. The direction of its operation will be clear from the context.

Circuit Encaps/Decaps sublayer

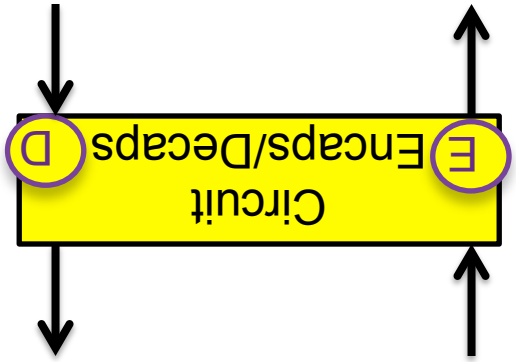


Addresses, Data, circuit_identifier

Addresses*, Data*

- The Circuit Encaps/Decaps sublayer has an interface on the upper SAP that includes a circuit_identifier parameter.
- It's lower interface does not have a circuit_identifier – the Circuit ID is encoded, somehow, in the Addresses or in the Data.

Circuit Encaps/Decaps sublayer

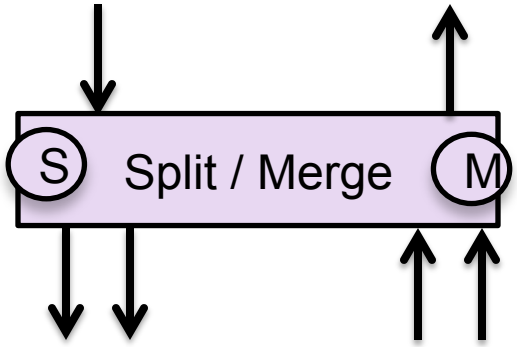


Addresses*, Data*

Addresses, Data, circuit_identifier

- In our diagrams, when we need the lower interface to have the circuit ID, we will turn the picture upside down.
- In this case, the augmented or altered addresses or data are on the upper interface, and the circuit_identifer on the lower.

Split / Merge sublayer

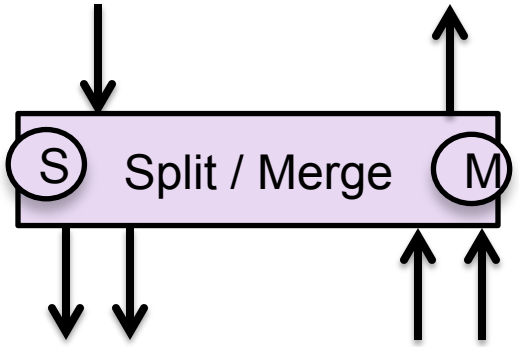


Addresses, Data, circuit_identifier

Addresses, Data, circuit_identifier

- The **Split** sublayer replicates packets, and may output packets with circuit_identifiers that are different each other, or different than the input.
- The Split sublayer may have one output (lower) port or more. If more than one, it sends one packet to each port.

Split / Merge sublayer

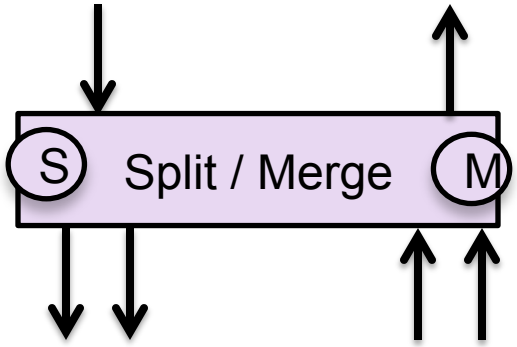


Addresses, Data, circuit_identifier

Addresses, Data, circuit_identifier

- The **Merge** sublayer combines the packets received on its one or more input (lower) ports, and outputs them on its upper port. **It does not discard frames.** That is done by the Sequence Discard sublayer.

Split / Merge sublayer

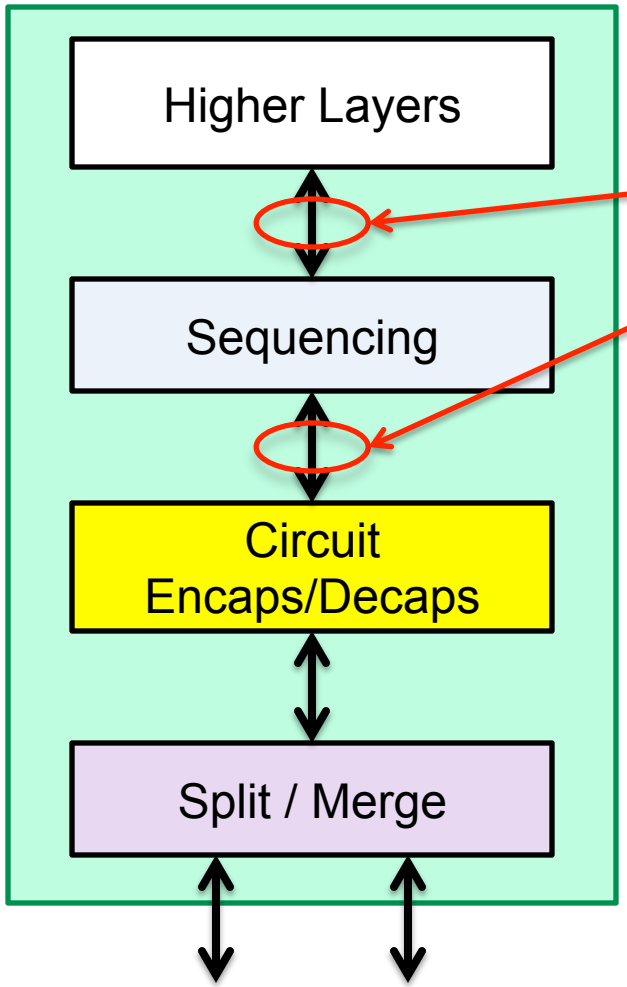


Addresses, Data, circuit_identifier

Addresses, Data, circuit_identifier

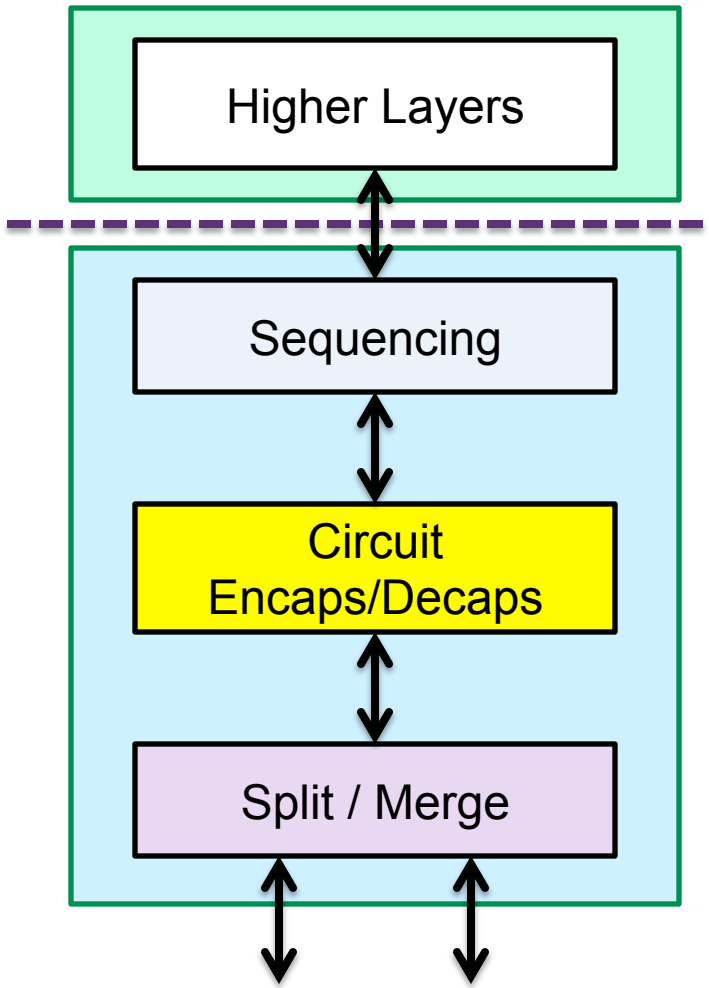
- If the Split / Merge sublayers **do not change the circuit_identifier**, then they work in an environment without that parameter.
- In that case, the Merge sublayer has very little to do; it passes packets transparently. And, if it has only one input (lower) port, it does absolutely nothing.

Single system



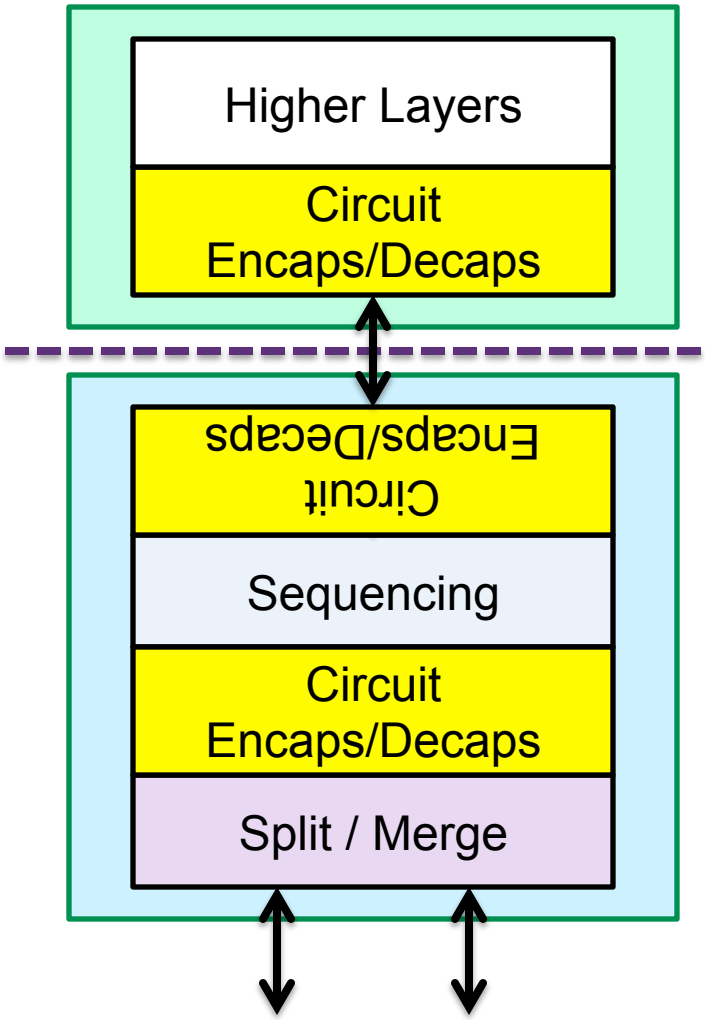
- Within a single system, the circuit_identifier can be **out**-of-band or **in**-band. Out-of-band methods include:
  - Socket ID.
  - Separate service instances.
  - Multiple Sequencing functions.

Multiple systems



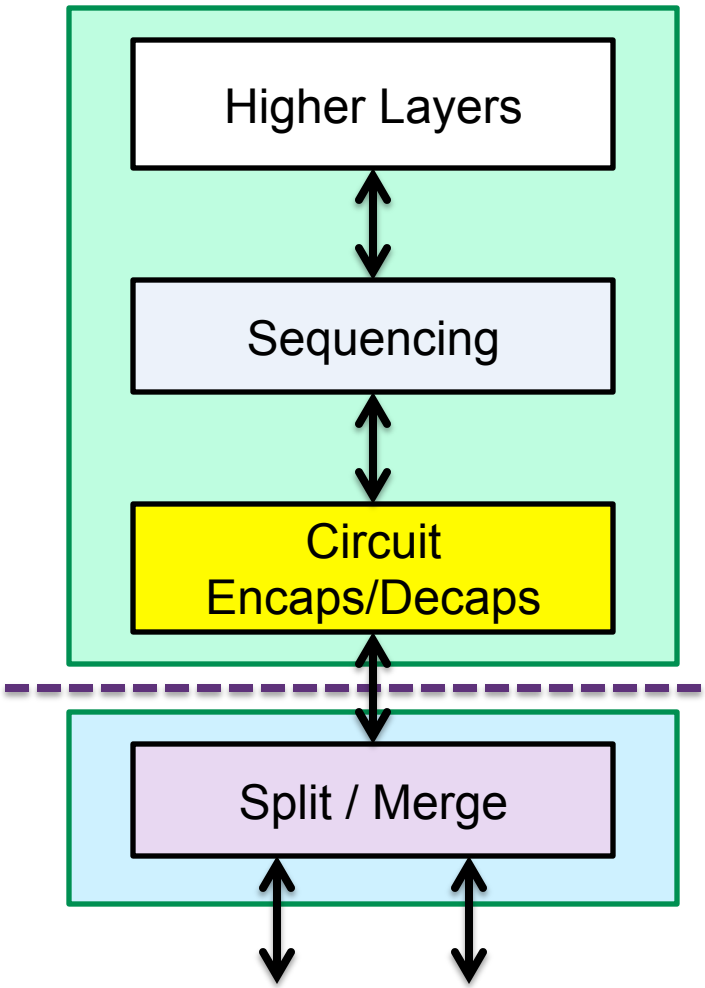
- For multiple systems, circuit identification must be **in-band**:
  - Some form of tag.
  - One or more layers of explicit addresses (e.g. VLAN ID or IP 5-tuple).
  - A circuit ID buried in an application.

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.

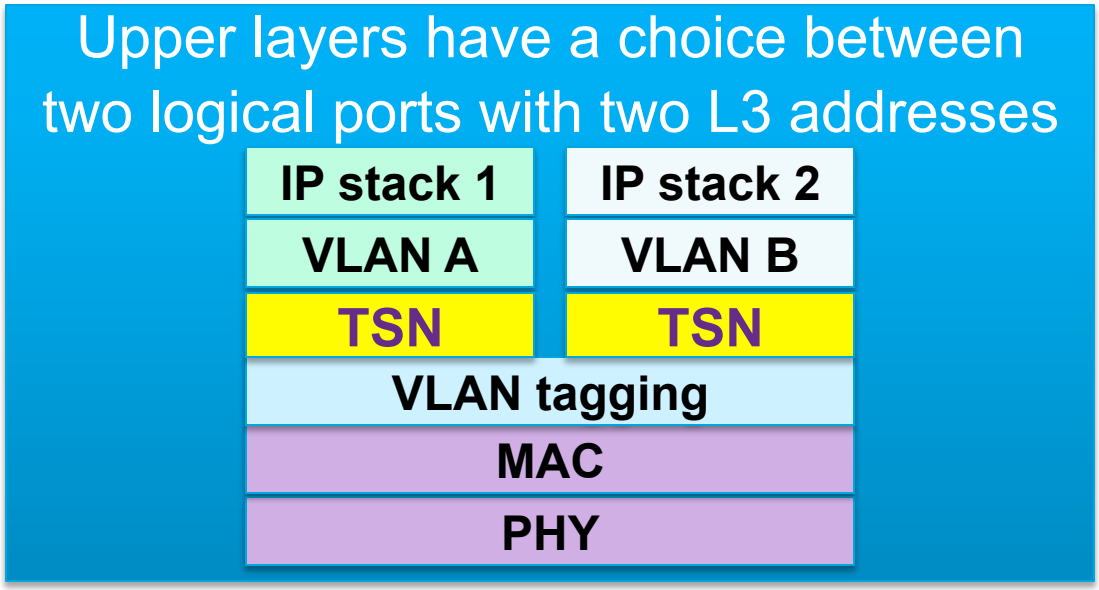
Multiple systems



- 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 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.

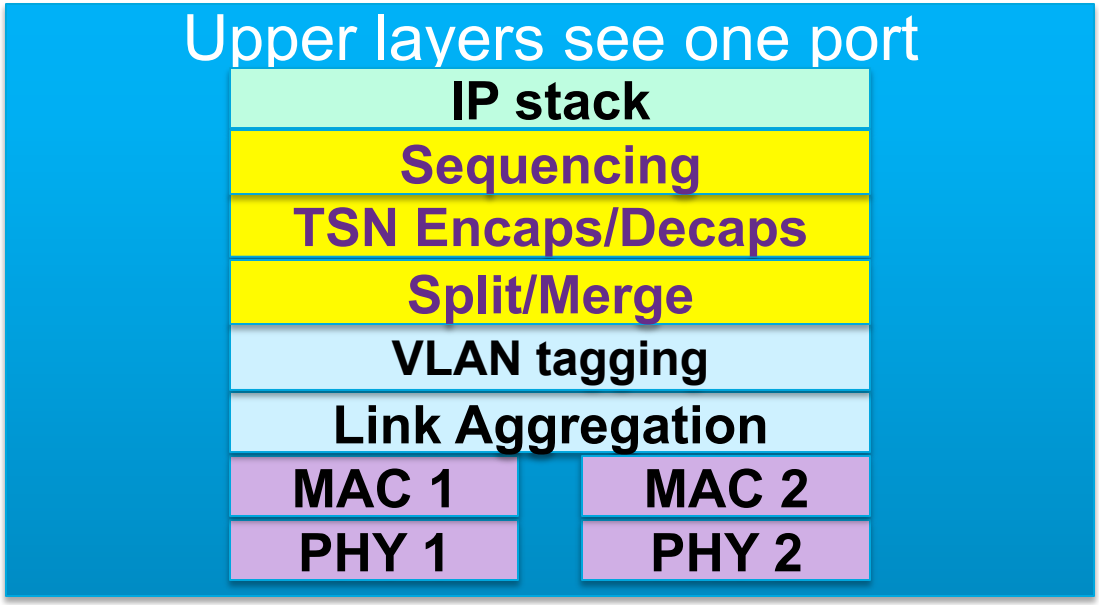


Single-port multiple VLAN host

- 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 layer splits regular traffic normally, and splits SR traffic using the circuit.

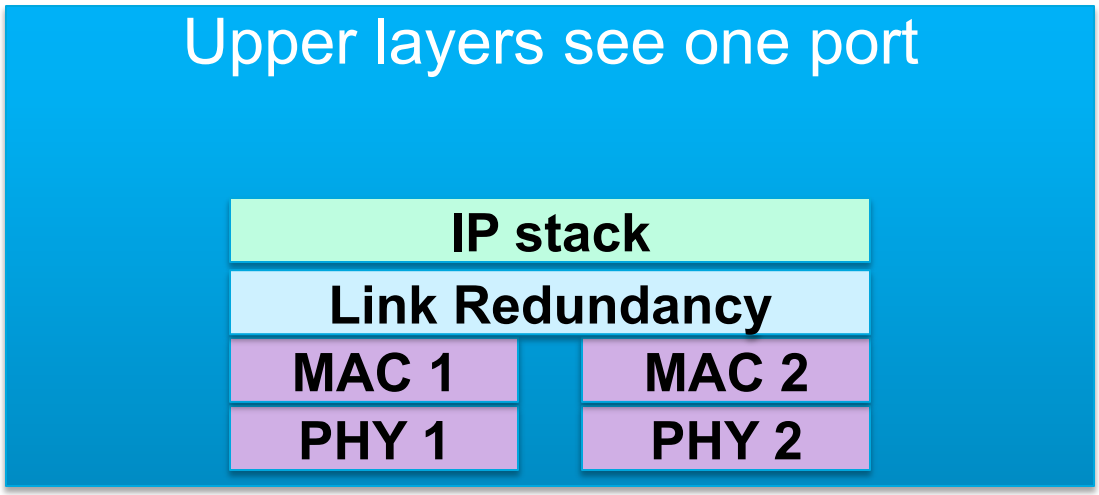


DRNI Host

- This diagram assumes that the Split/Merge function duplicates frames, and passes to the Link Aggregation layer (by circuit ID, either explicitly in the frames or by unspecified means) 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.

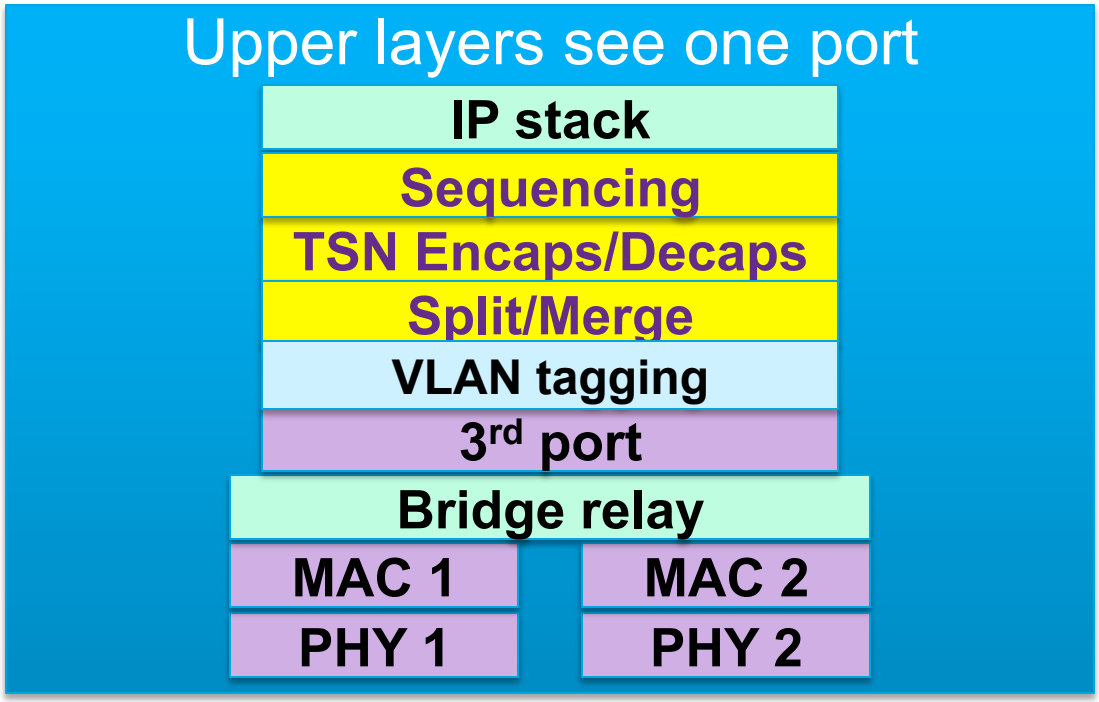


Dual-port host (HSR or PRP)

- 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.

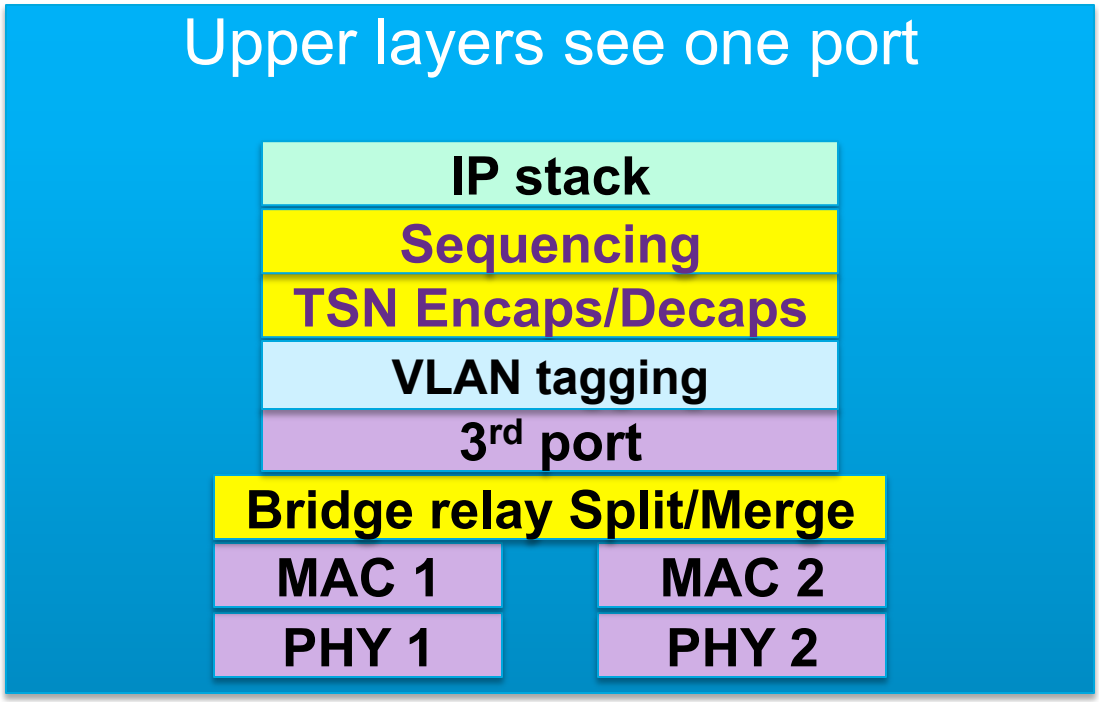


Dual-port relay host (bridged)

- 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 differently-labeled circuits, and the bridge part directs them on different paths.



Dual-port relay host (bridged)

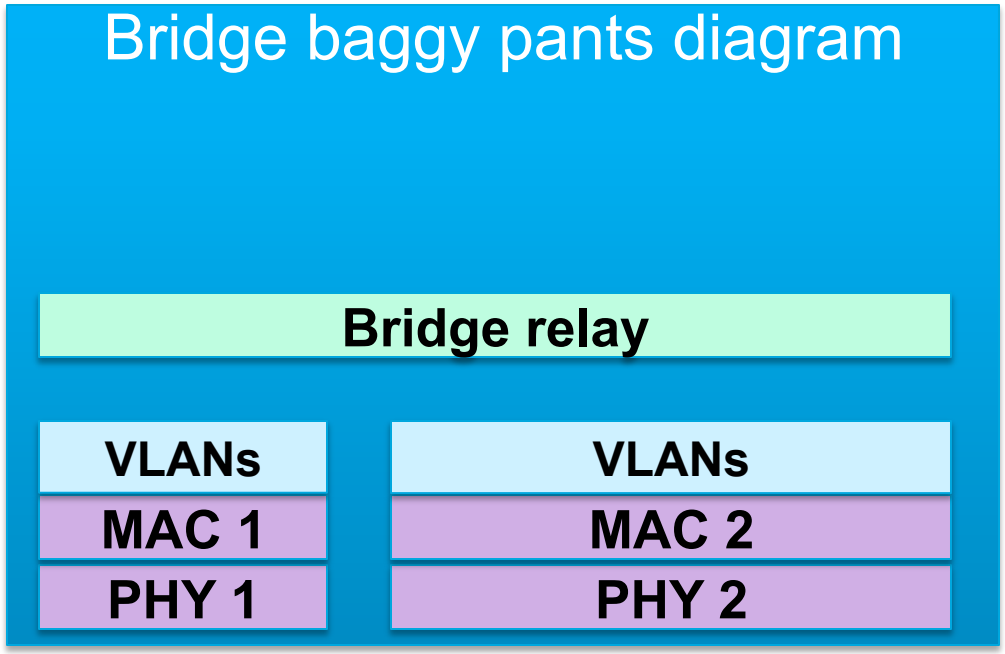
- In the preceding diagram, we do not change the circuit IDs in the host stack, and output a single frame to the Bridge.
- The standard Bridge Relay serves the functions of a Split/Merge function.

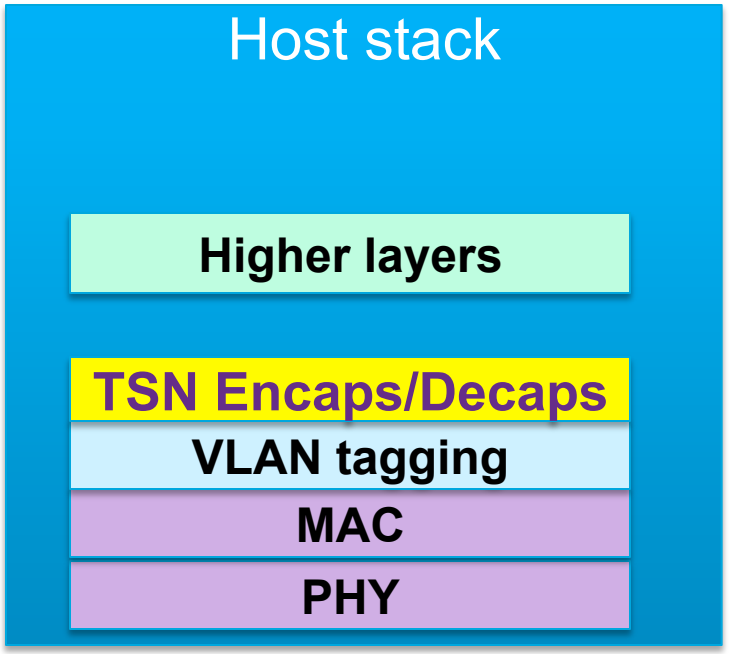
Dual-port relay host (bridged)

- 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

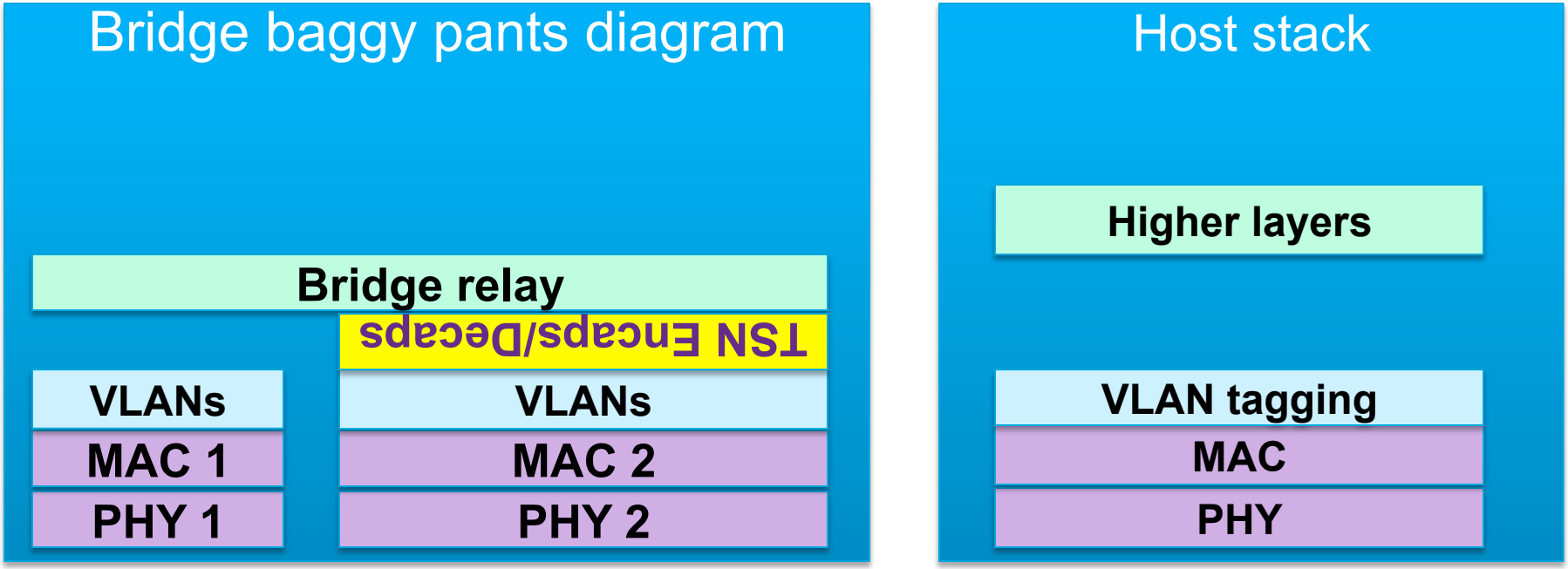
- A **Bridge** may want to offer these services to an end station that is TSN unaware.
- What you start with:





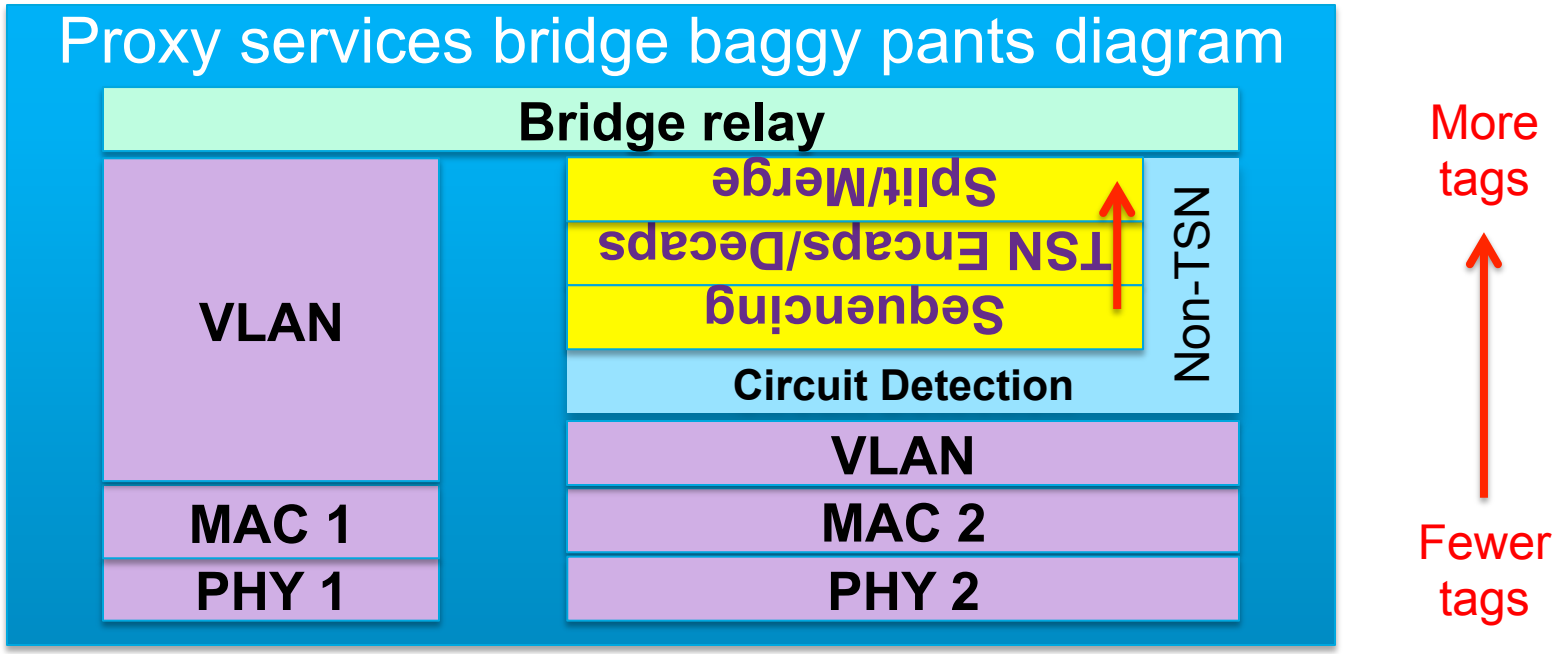
Proxy bridge stack

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



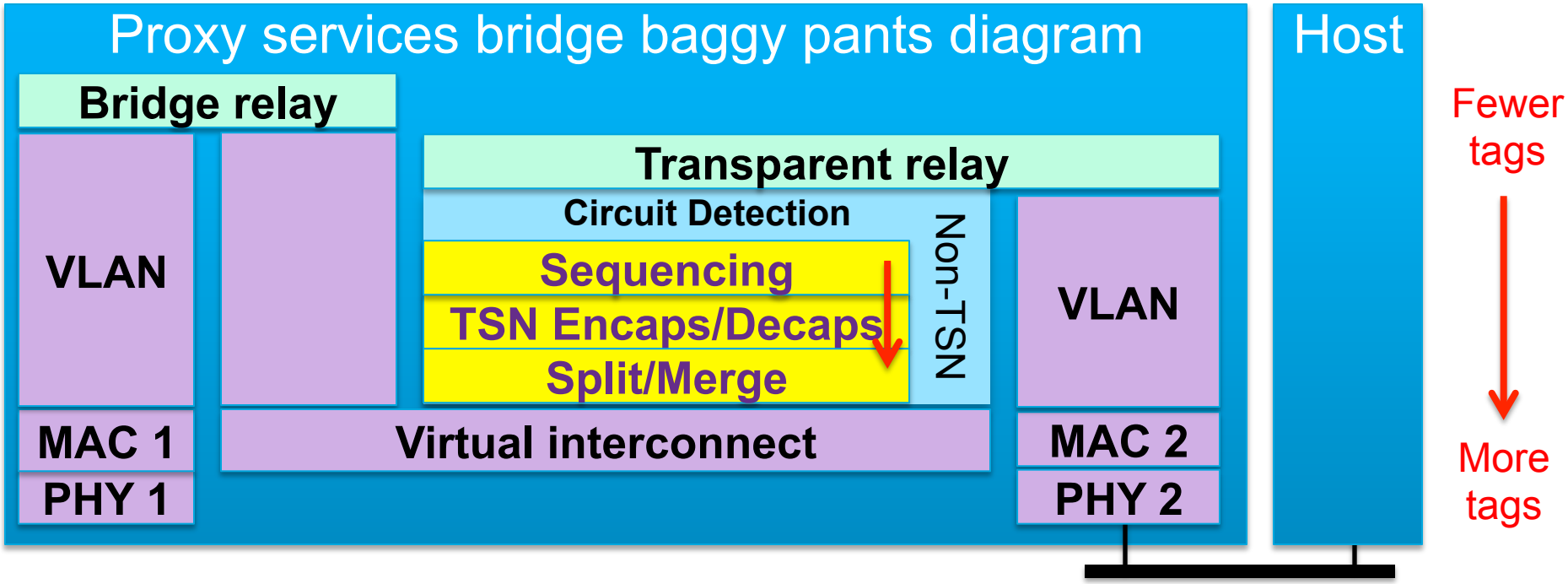
Proxy bridge stack

- A Circuit Detection sublayer is then required to convert something explicitly in the data frame to a circuit_identifier:



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

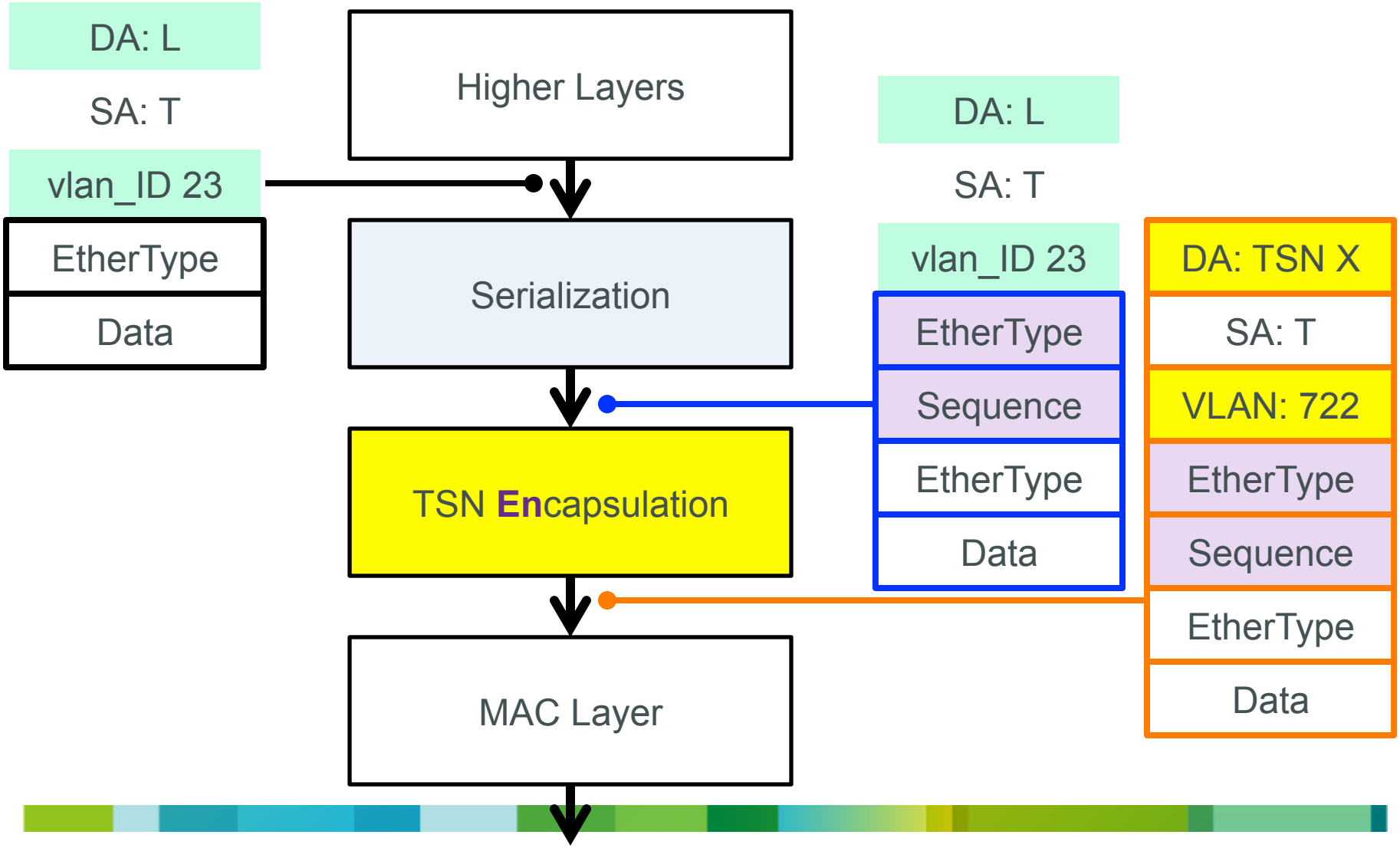


Alternative TSN Encapsulations

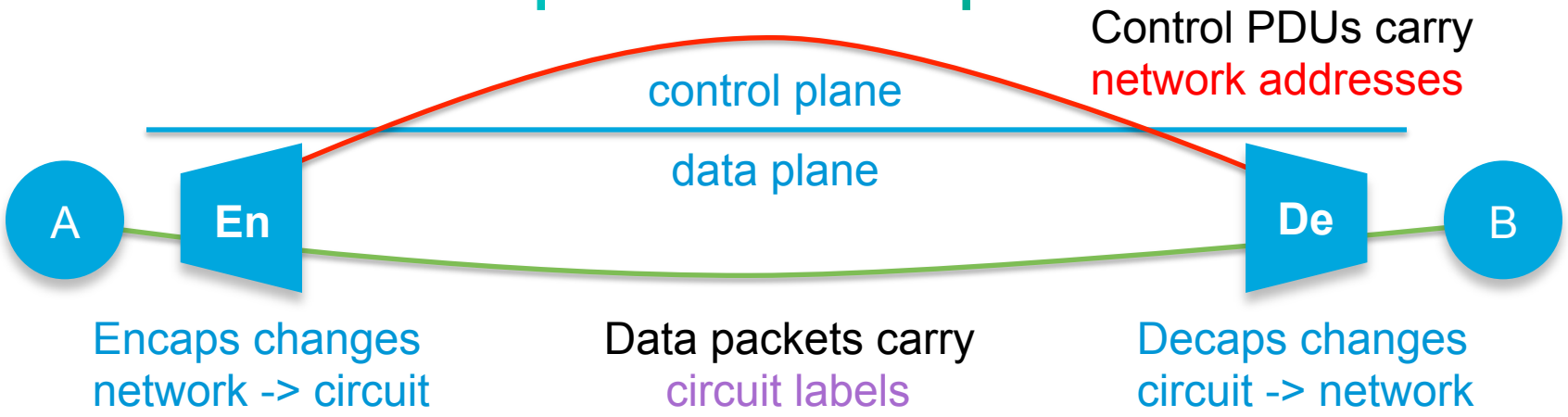
1. **Sequenced TSN** encapsulation

- 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

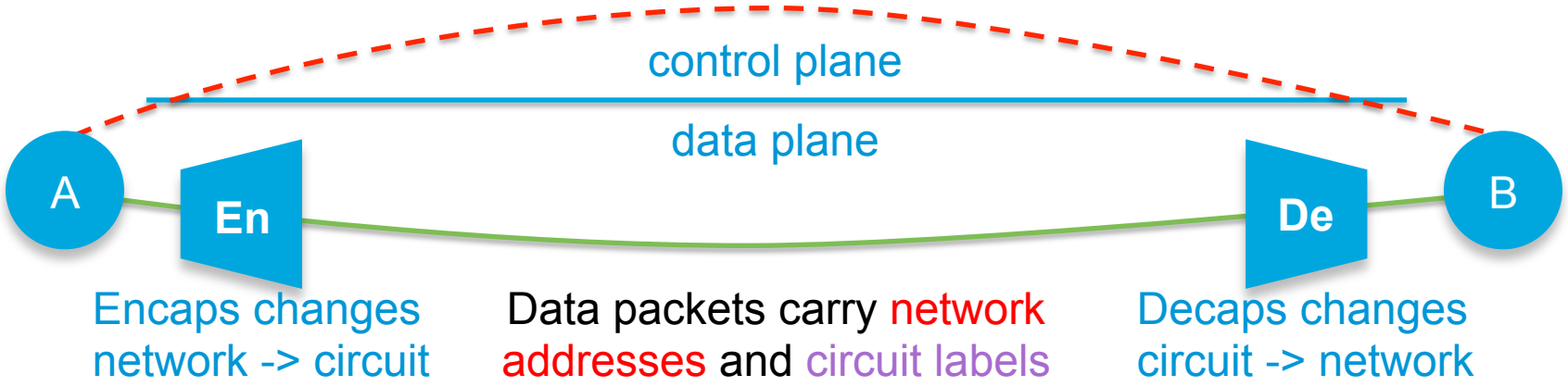


Are other encapsulations possible? Yes!



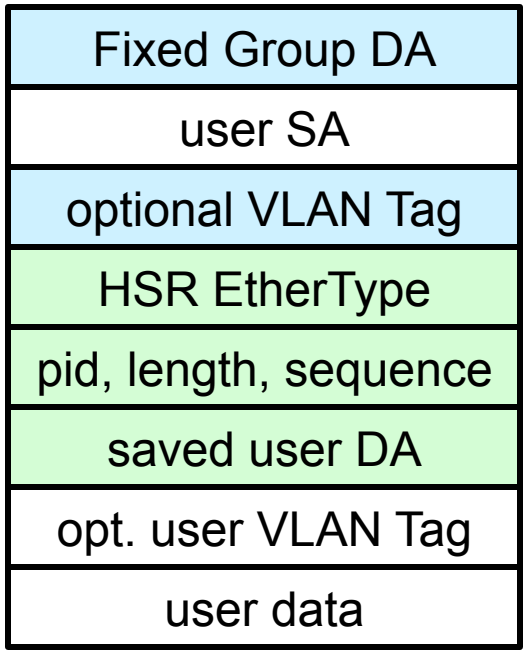
- To use the current AVB format, while making TSN services transparent to the user, the Encapsulation function destroys information, and the Decapsulation function restores it.
- Let's call this **out-of-band tunneling**.

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

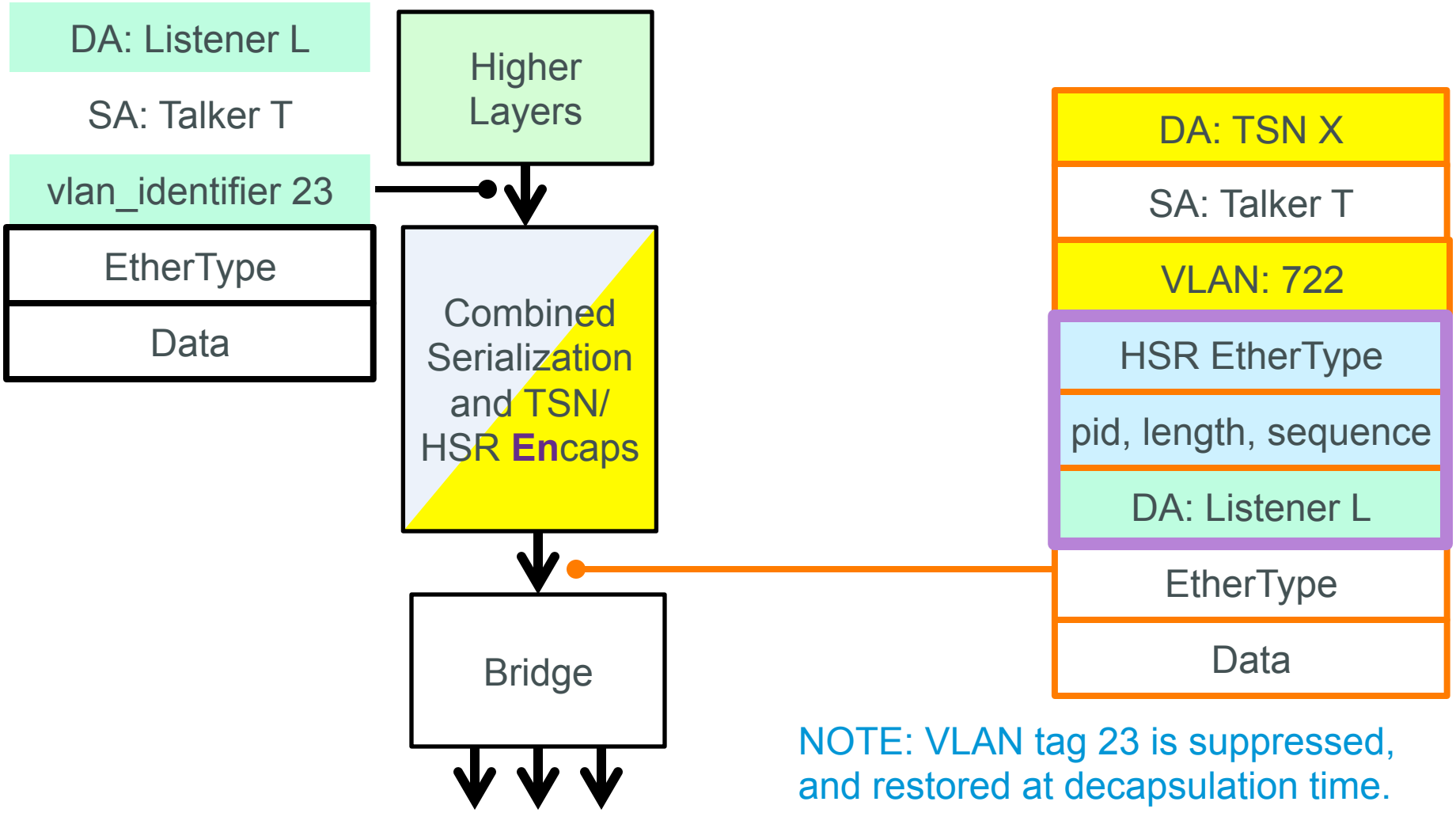


- The **IEC 62439-3 High-Speed Seamless Redundancy (HSR)** encapsulation provides in-band tunneling.
- Almost. On the good side:
  - HSR encapsulates the original destination MAC and VLAN, rather than using the data plane.
  - HSR includes a sequence number for seamless redundancy. (That **is** the name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

2. HSR**-like** seamless redundancy

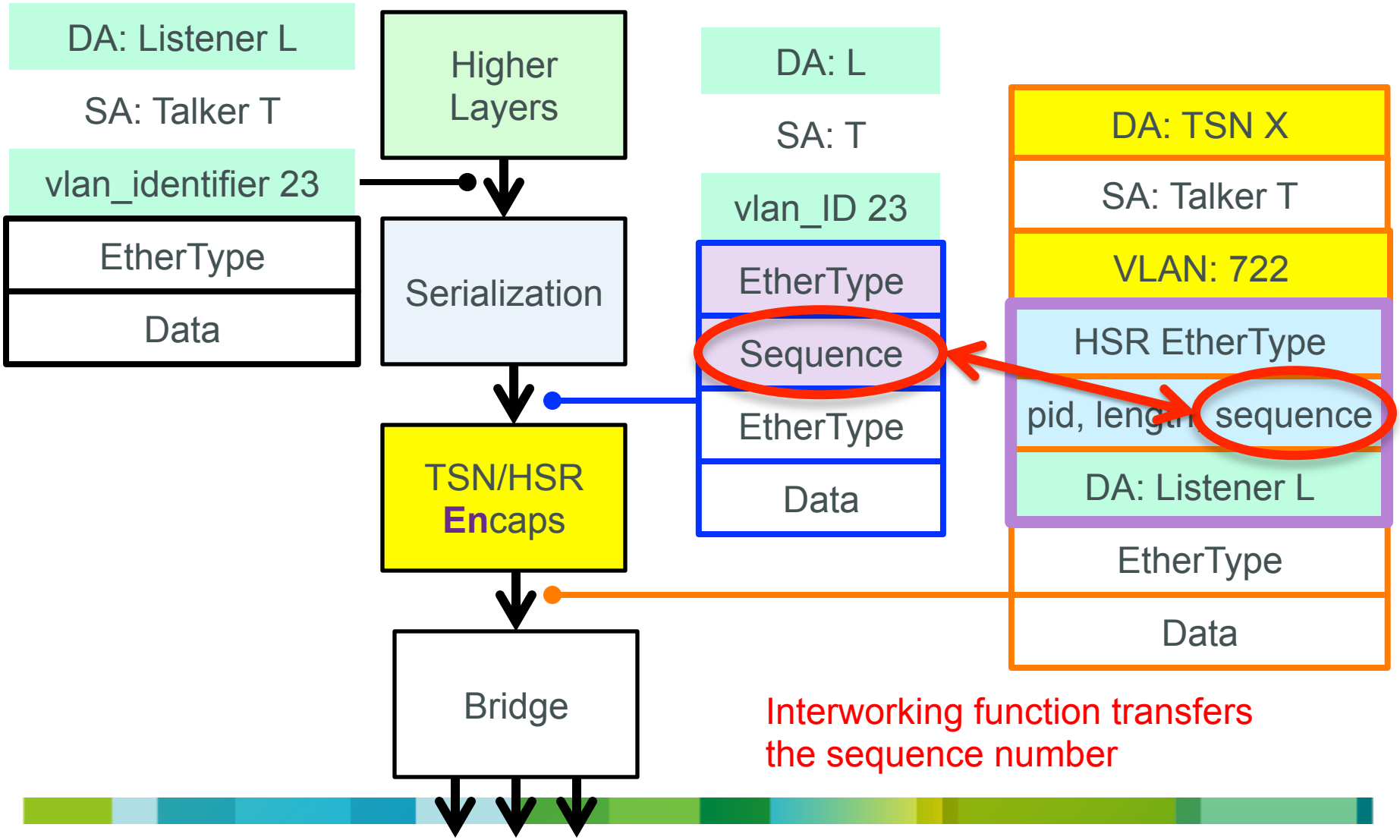
- But! HSR only works if the network is an HSR ring, because the TSN circuit address has been buried behind the fixed HSR DA.
- To make HSR work over 802.1 networks:
  - Change to the PDU format:
    - We substitute the TSN circuit DA for the HSR fixed DA.
  - Changes to the use of that format:
    -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

2. HSR seamless redundancy (option 1)

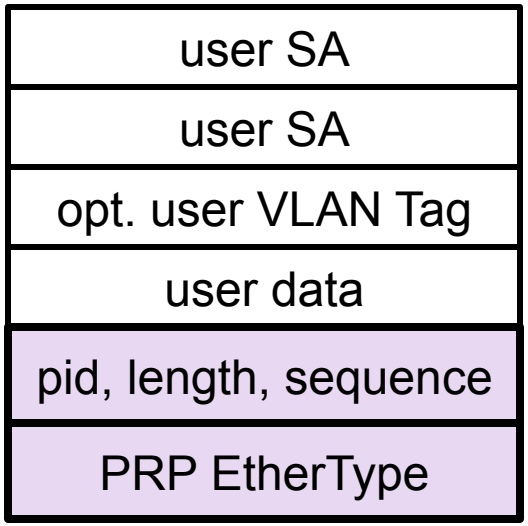


NOTE: VLAN tag 23 is suppressed, and restored at decapsulation time.

2. HSR seamless redundancy (option 2)

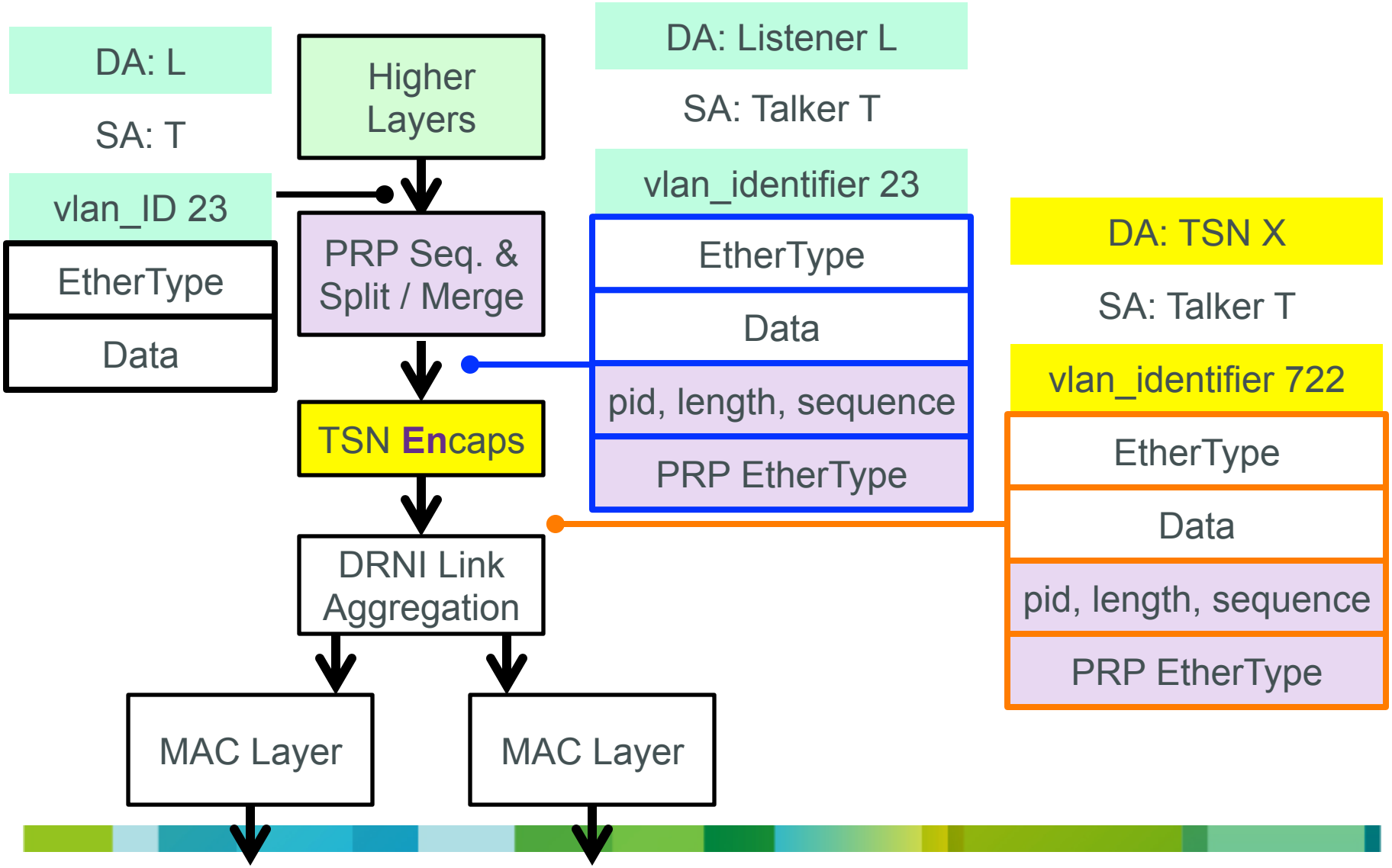


3. PRP seamless redundancy + TSN



- 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



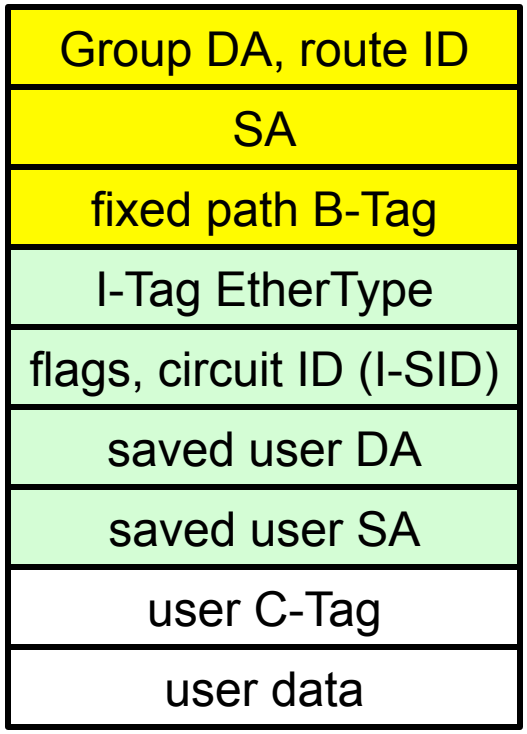
3. IEC 62439-3 PRP + TSN

- 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.
- A DRNI / Link Aggregation unit, as described earlier, would connect this stack to the physical ports.

3. IEC 62439-3 PRP + TSN

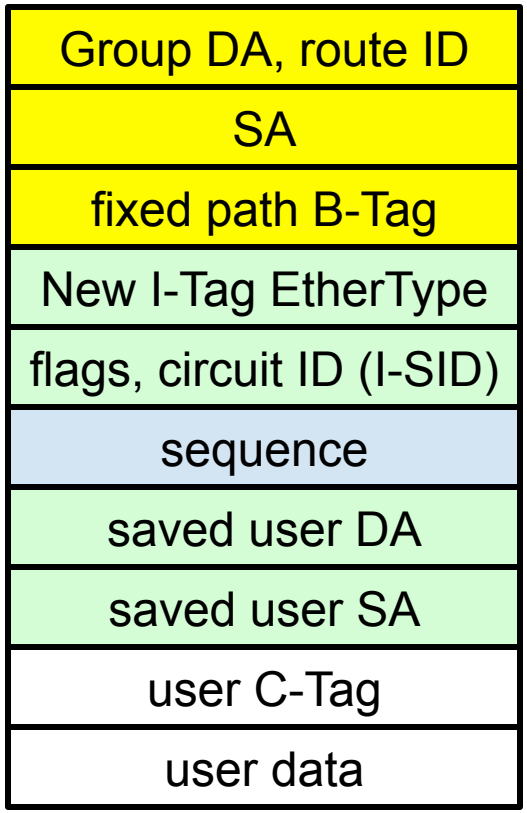
- You can think of this formulation as decomposing the “PRP Link Redundancy” sublayer defined in IEC 62439-3 into a “PRP Sequencing and Split/Merge” sublayer, a TSN encapsulation sublayer, and a DRNI sublayer, that together accomplish the same goal.
- The reader may also notice that, when a trailer is used, the ordering of the layers becomes ambiguous.

4. PBB-TE for TSN



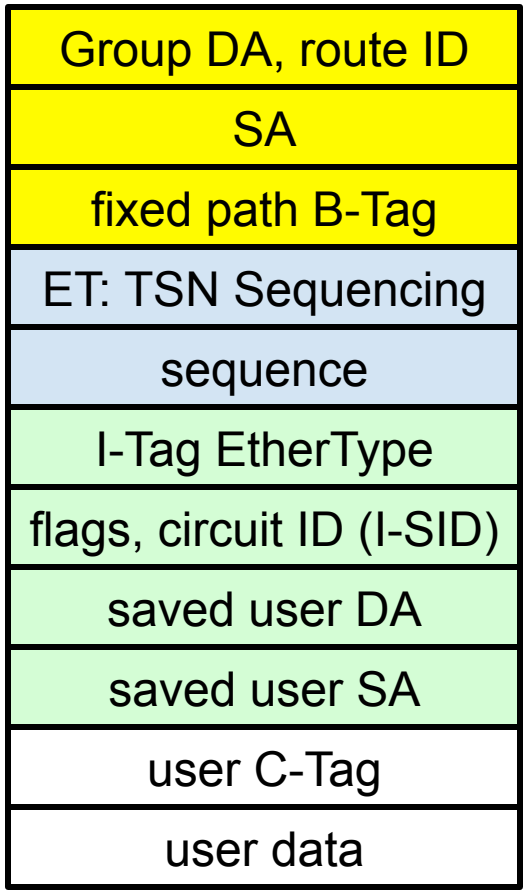
- MAC-in-MAC (802.1ah) solves the circuit ID and fixed path problems using in-band tunneling.
  - It encapsulates the data now carried in the control plane.
  - But, it has no circuit ID or sequence number for seamless redundancy.

4. PBB-TE for seamless redundancy (1)



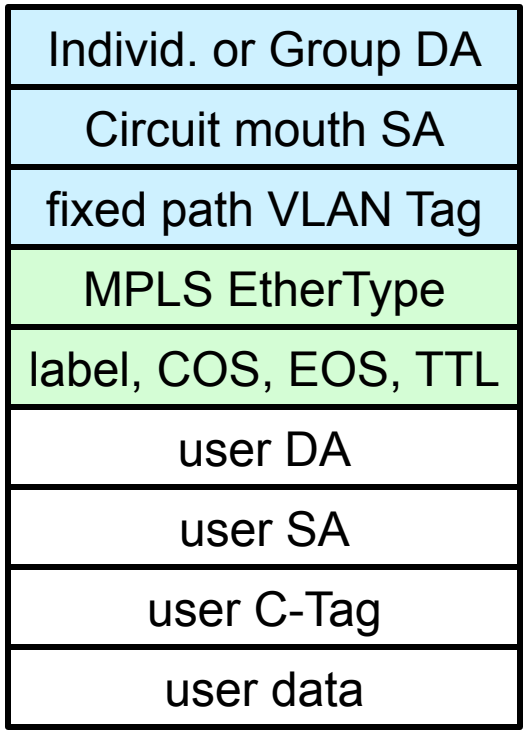
- We must either have an additional tag for the sequence number (not clear where), or a new I-Tag format that includes a sequence number (shown at left).
- Or, just use it for tunneling.

4. PBB-TE for seamless redundancy (2)



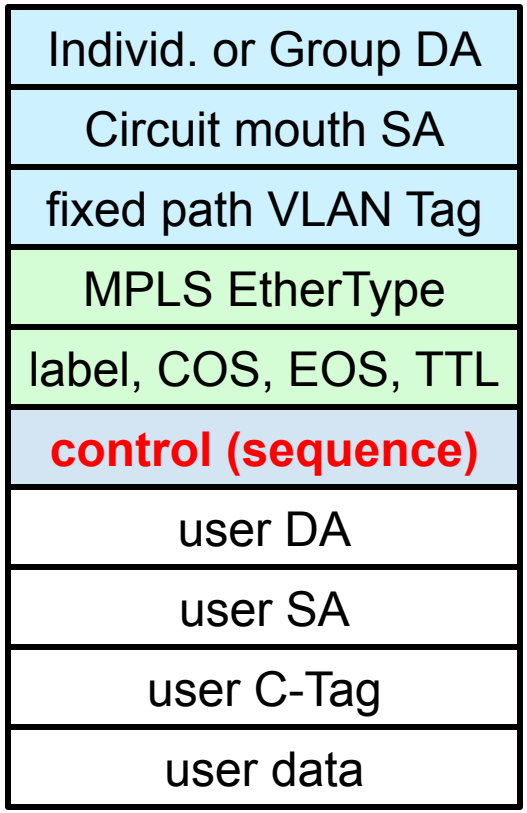
- We must either have an additional tag for the sequence number (shown at left), or a new I-Tag format that includes a sequence number.
- Or, just use it for tunneling.

5. MPLS Ethernet Pseudowire



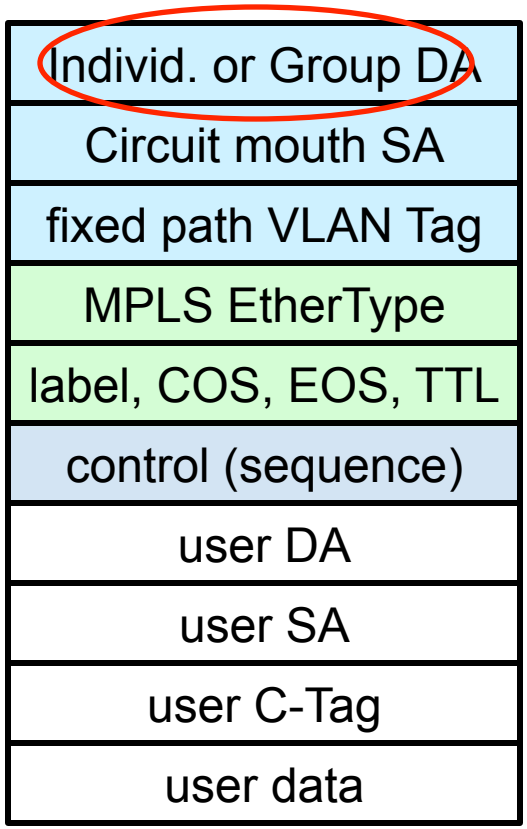
- An **MPLS pseudowire** provides almost exactly the same encapsulation structure as 802.1ah.
- Like the current AVB encapsulation, it provides out-of-band tunneling.

5. Pseudowires for seamless redundancy



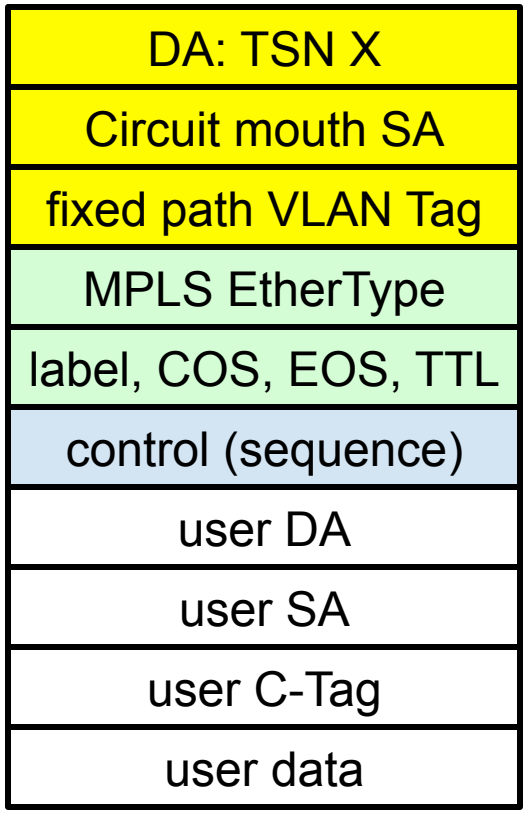
- Plus, Pseudowires have an optional control word that provides a **sequence number for seamless redundancy**.

5. MPLS Ethernet Pseudowire



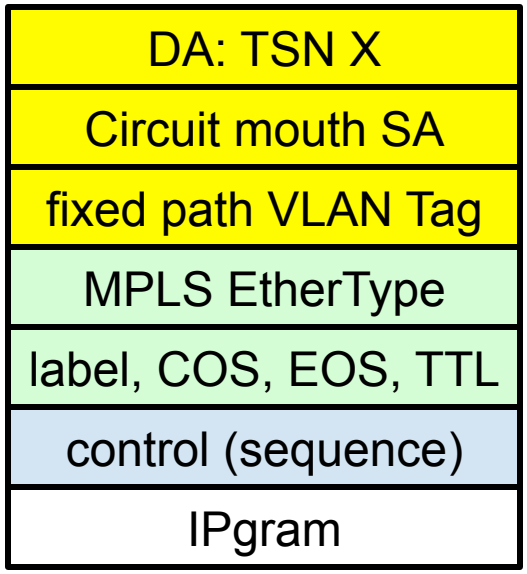
- There is one **problem**: MPLS uses either the unicast next-hop destination, or a fixed Group DA.

5. MPLS Ethernet Pseudowire



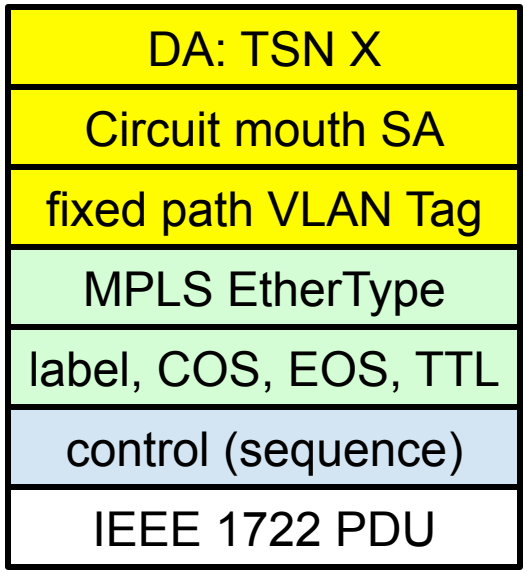
- But, we just talked about the TSN Encaps/Decaps layer changing the outer MAC address!
- So, this is the encapsulation we would use.

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 very 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.

7. 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**. 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.
- Since only the latter works, it is not clear that there is any value in encapsulating and carrying MAC addresses and/or VLAN IDs, as opposed to swapping the them in and out in the TSN Encaps/Decaps layer.
- (L2 encapsulations are useful! But, not as a TSN format.)

Summary

Summary 1/2

- When you get the layering right, 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 proper layering.
- We should define the Sequencing, TSN Encaps/Decaps, and Split/Merge sublayers.

Summary 2/2

- When you get the layering right, several existing protocols support TSN circuits.
  - TSN, HSR, MPLS, PBB-TE, or no protocol at all.
  - There are more.
- When you get the layering right, several existing protocols support seamless redundancy, already.
  - HSR, PRP + TSN, Ethernet pseudowires, IPgram pseudowires.
  - There are more.
- There is little point in, and there are definite disadvantages to, encapsulating MAC addresses or VLAN tags inside the TSN encapsulation.

Thank you.