A Day in the Life of an L2/L3 TSN Data Packet.

Norman Finn
Version 1

Feb. 17, 2014

This presentation

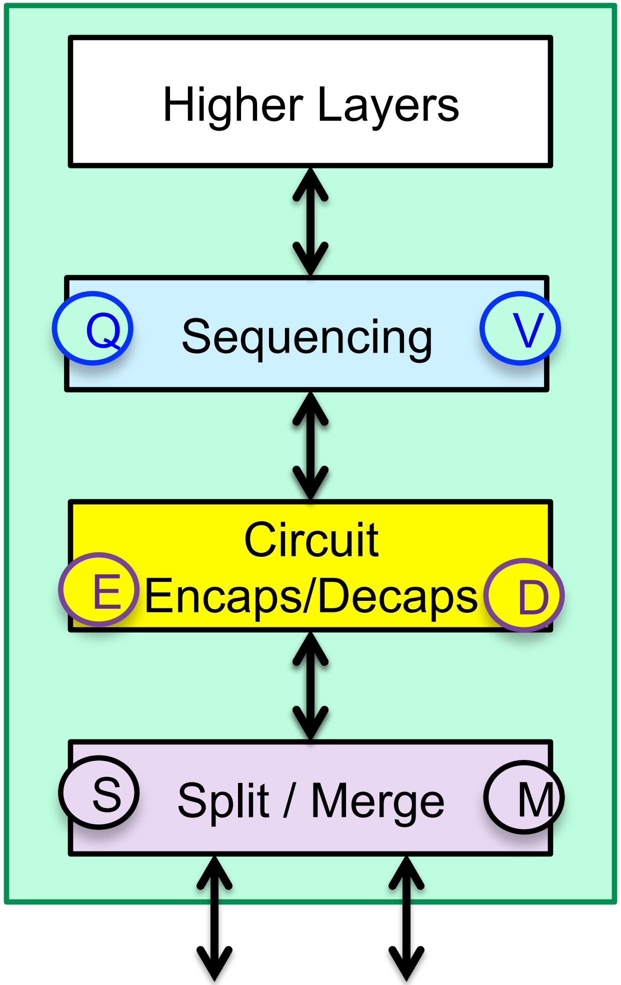
- This presentation, tsn-nfinn-Day-In-The-Life-0214-v01 is an annex to a two-part presentation.
- Part 1, tsn-nfinn-L2-Data-Plane-0214-v03, introduces concepts on which these presentations depend.
- Part 2, tsn-nfinn-L3-Data-Plane-0214-v02, is concerned with Layer 3 issues.
- See also cb-nfinn-How-Many-VLANs-0214-v01.

Outline

1. A very brief introduction, using concepts introduced in the preceding decks.
2. Case 1: A "day in the life of a packet" for an end-to-end Ethernet Bridged LAN with seamless redundancy.
3. Case 2: The same for a mixed L2/L3 network, along with an interlude about interworking functions, and alternative scenarios.
4. Case 3: Describes the use of a possible circuit encapsulation using IEC 62439-3 HSR or PRP.
5. Case 4: Looks at other possibilities.
6. A one-slide summary of conclusions is given.

Layering

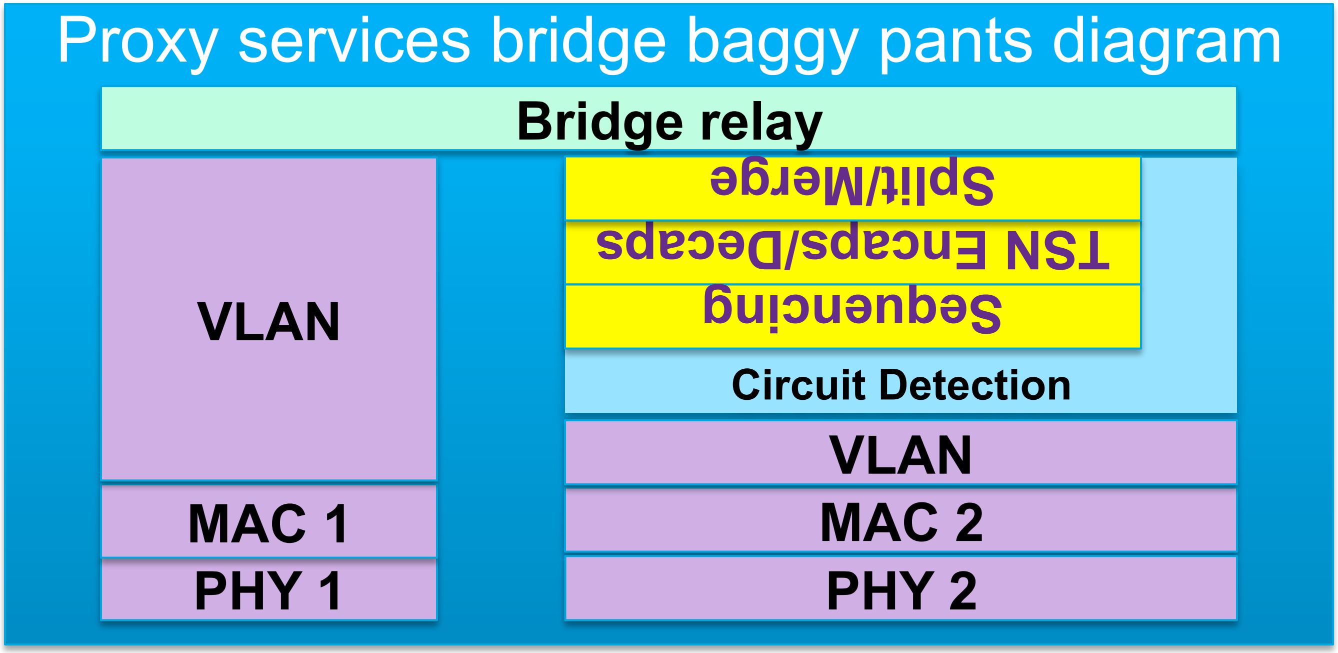
Layer reminder (from L2-Data-Plane)



- Higher Layers work as always.
- Sequencing numbers packets Q, and discards duplicates V (V includes X)
- Circuit Encaps E /Decaps D marks individual circuits.
- Split S /Merge M have one circuit ID above and two below its layer. It uses sequence numbers for “fools paradise” checking.

Proxy bridge stack (from L2-Data-Plane)

- This is the stack for a bridge that proxies for a non-TSN client, e.g. Bridge 8 in the following examples:



Proxy services bridge baggy pants diagram

Bridge relay

VLAN

MAC 1

PHY 1

Split/Merge

TSN Encaps/Decaps

Sequencing

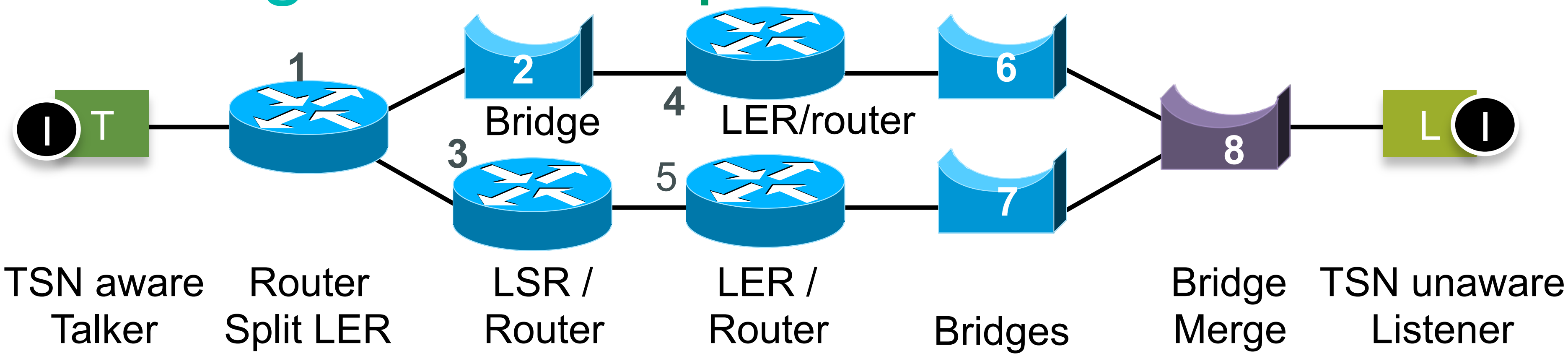
Circuit Detection

VLAN

MAC 2

PHY 2

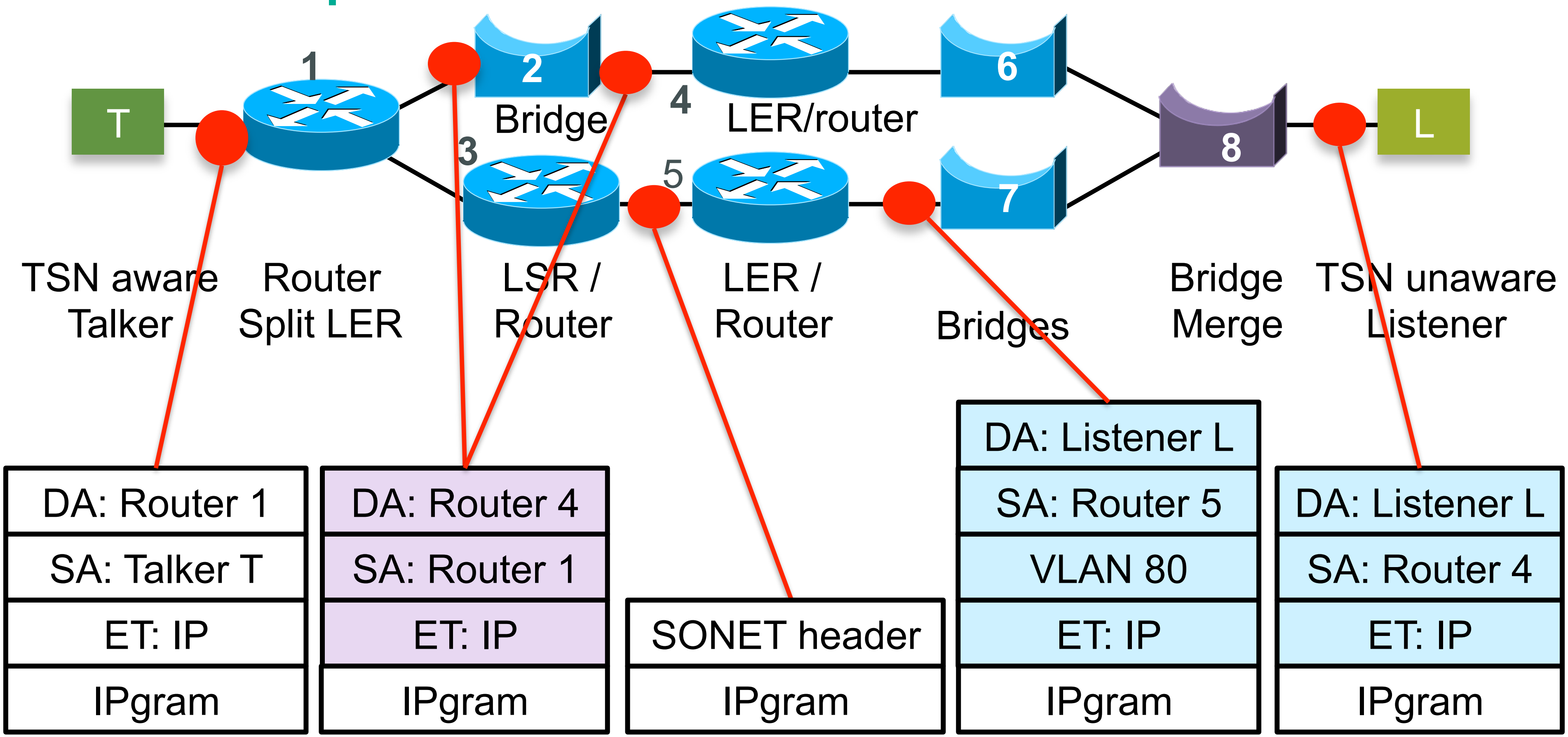
Peering relationships



Talker and Listener are **peers** at some layer.

- Let’s say they exchange IPgrams, either IPv4 or IPv6.

Natural packets



DA: Router 1
SA: Talker T
ET: IP
IPgram

DA: Router 4
SA: Router 1
ET: IP
IPgram

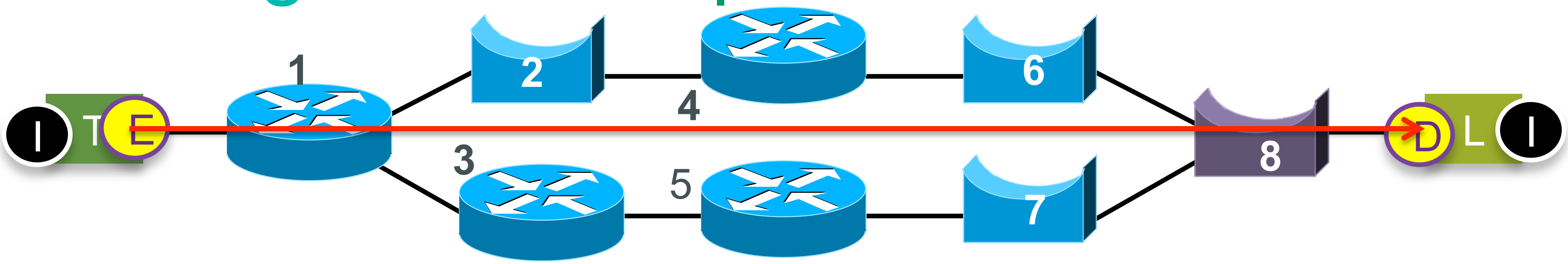
SONET header
IPgram

DA: Listener L
SA: Router 5
VLAN 80
ET: IP
IPgram

DA: Listener L
SA: Router 4
ET: IP
IPgram

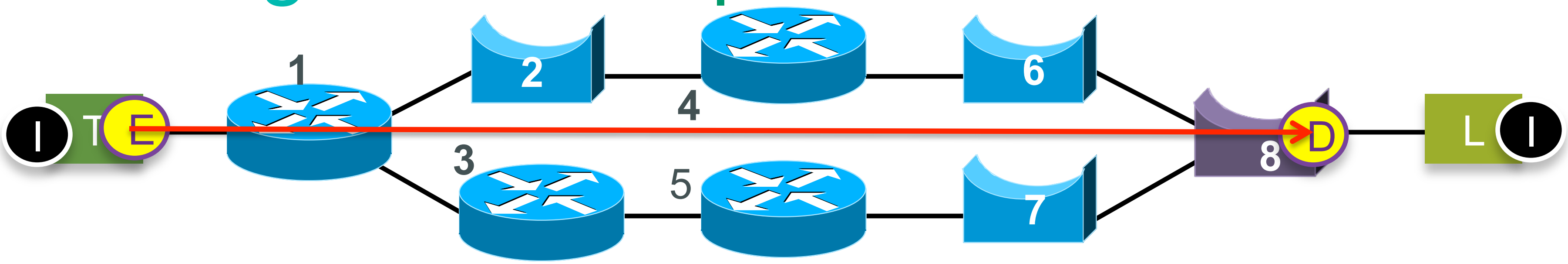
- **Without TSN** routers route and bridges bridge.

Peering relationships



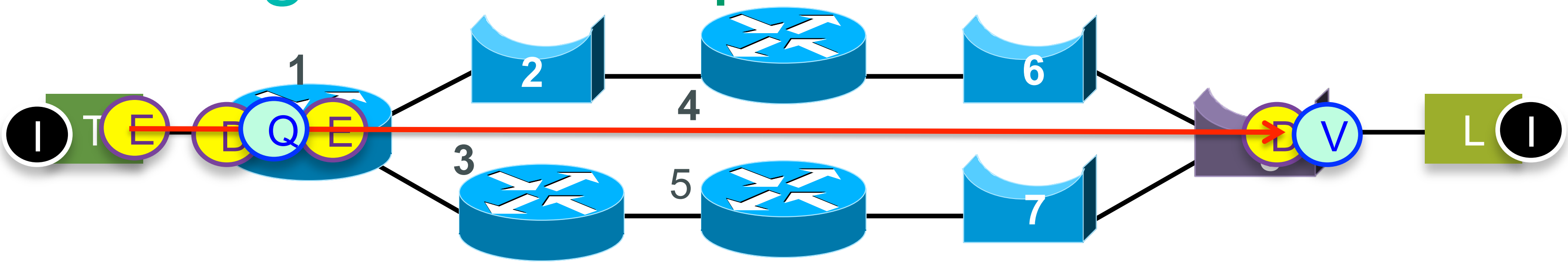
- The operator wants Talker T and Listener L to have a TSN **circuit** relationship, so that they can get the special TSN Qualities of Service.

Peering relationships



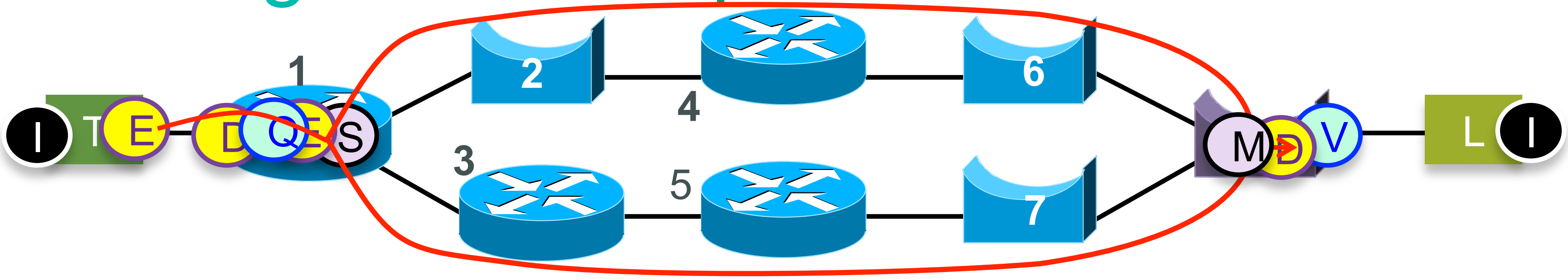
- But, the Listener is TSN-unaware, so Bridge 8 has to provide the TSN Circuit Decaps (D) as a **proxy service**.

Peering relationships



- Furthermore, the operator wants to provide the Sequencing function Q , V . But, in this example, Talker T does not know about sequencing.
- So, **Router 1 proxies** for Talker 1. Since Listener L is TSN-unaware, Bridge 8 removes the sequence numbers.

Peering relationships



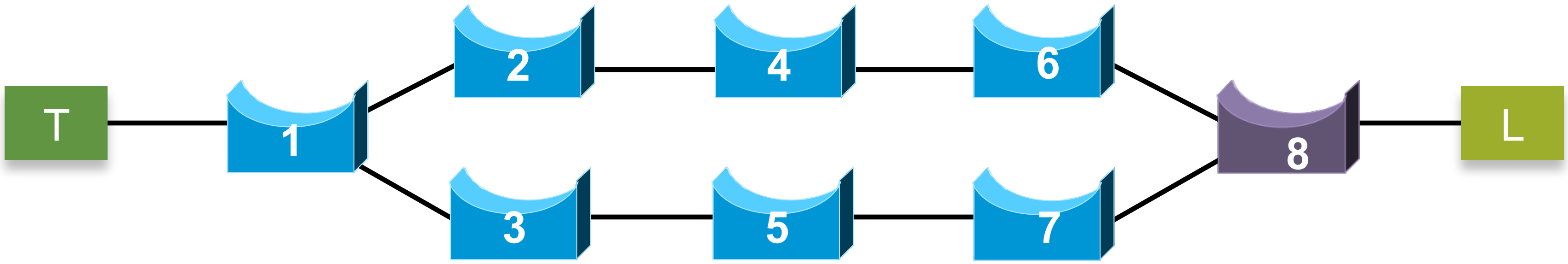
- Also, the operator wants seamless redundancy, to protect the packets better.
- This requires a Split function Ⓢ and a Merge function Ⓜ , both at the points where the **circuit** bifurcates.

Case 1:
Layer 2 end-to-end
Sequenced TSN tagging

Sequenced TSN tagging

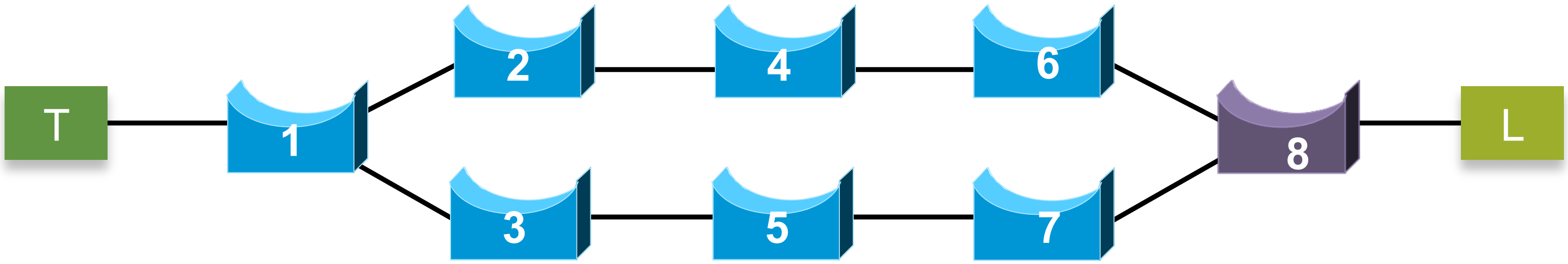
- Top-down view

Layer 2 only: **TSN tagging**



- Given that introduction, let us examine the simplest case: end-to-end connectivity through a Bridged LAN.

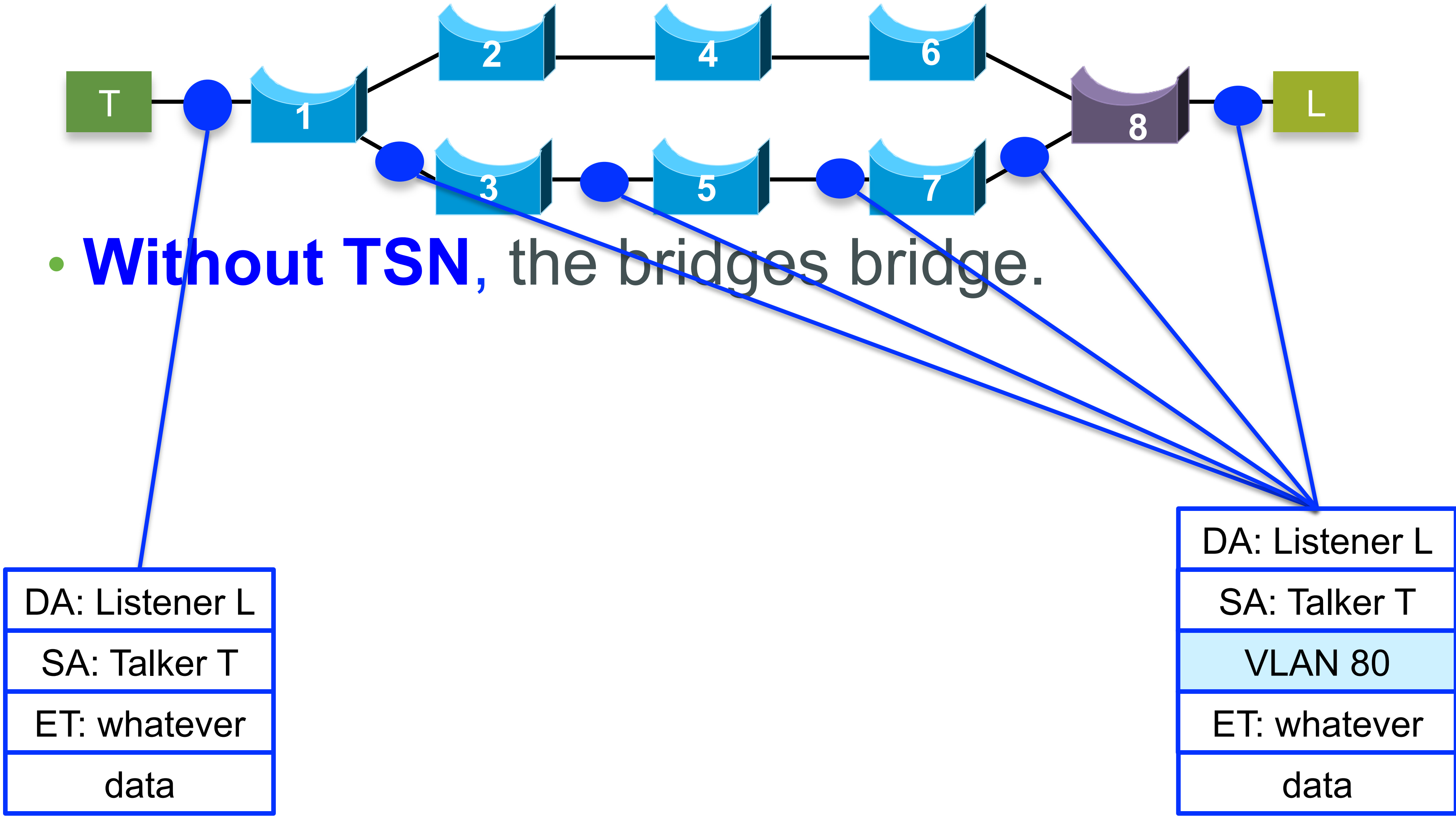
Layer 2 only: **TSN tagging**



- Talker is TSN-aware, Listener is not.
- Talker is **not** VLAN-aware, Listener **is** VLAN-aware.

Natural packets

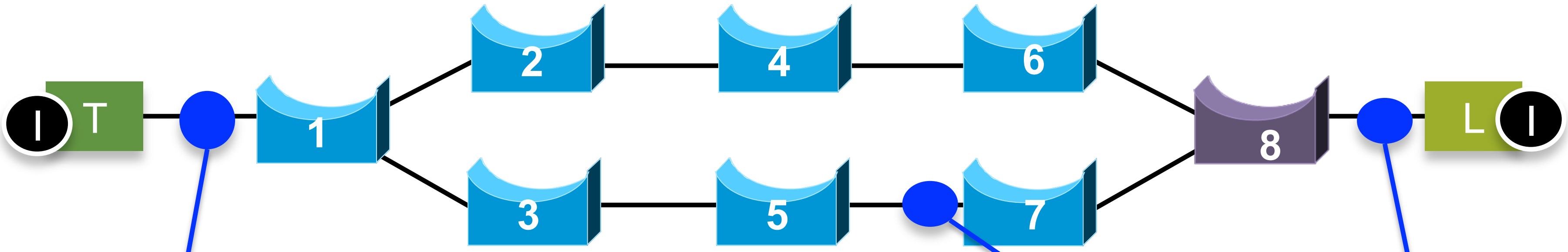
- **Without TSN**, the bridges bridge.



DA: Listener L
SA: Talker T
ET: whatever
data

DA: Listener L
SA: Talker T
VLAN 80
ET: whatever
data

Peering relationships



- Talker T and Listener L have a higher layer relationship.

DA: Listener L

SA: Talker T

ET: whatever

data

DA: Listener L

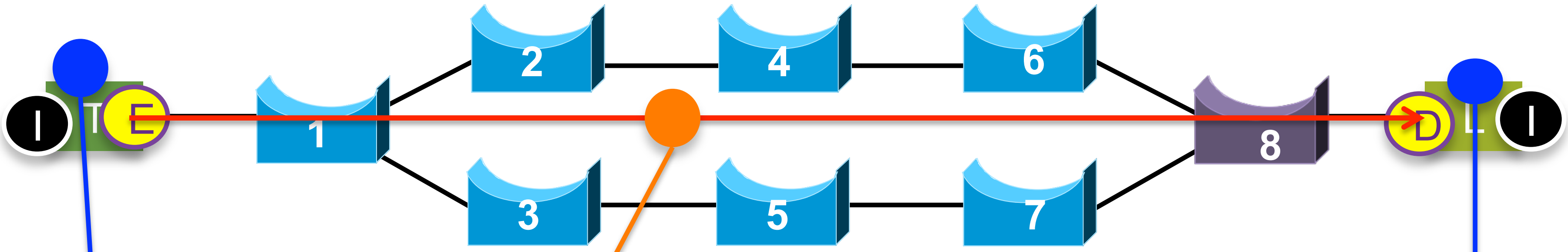
SA: Talker T

VLAN 80

ET: whatever

data

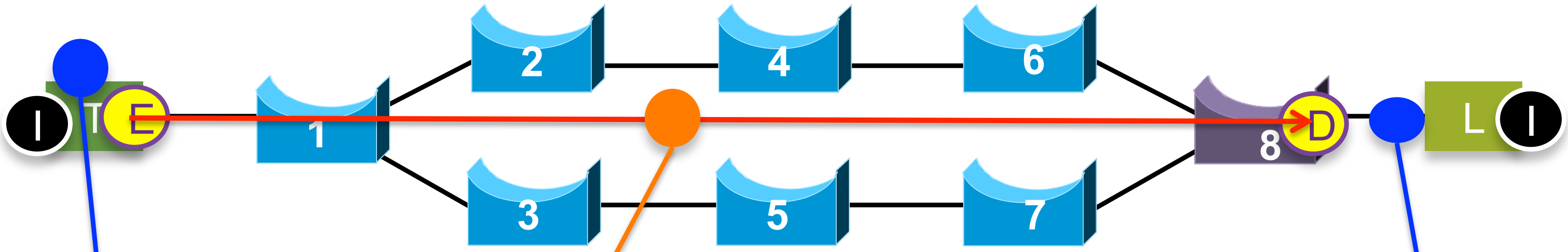
Peering relationships



- The operator wants Talker T and Listener L to have a TSN **circuit** relationship E, D, (734[99]) so that they can get the TSN QoS. (The bridges need the circuit ID in order to provide the TSN QoS.)

DA: Listener L	DA: TSN 734	DA: Listener L
SA: Talker T	SA: T	SA: Talker T
circuit_ID	VLAN tag 99	VLAN 80
ET: whatever	ET: whatever	ET: whatever
data	data	data

Peering relationships



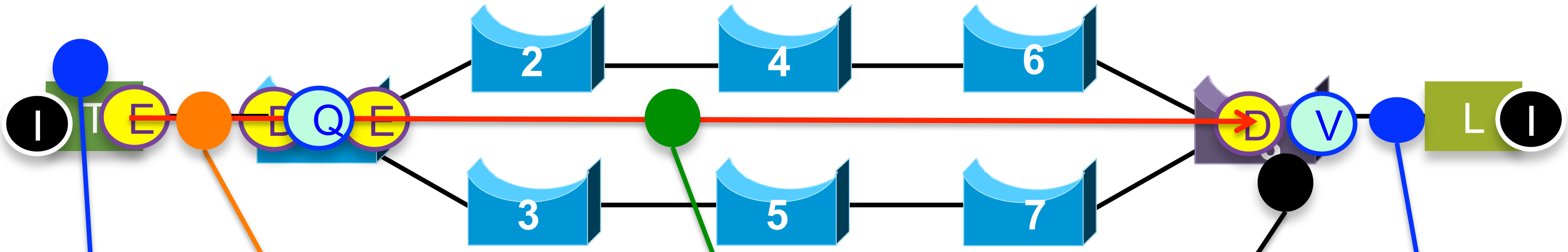
- But, the Listener is TSN-unaware, so Bridge 8 has to provide the TSN Circuit Decaps D as a **proxy service**.

DA: Listener L
SA: Talker T
circuit_ID
ET: whatever
data

DA: TSN 734
SA: T
VLAN tag 99
ET: whatever
data

DA: Listener L
SA: Talker T
VLAN 80
ET: whatever
data

Peering relationships



- The operator also wants the **Sequencing function** Q V , proxied T ans L by Router 1 and Bridge 8.

DA: Listener L
SA: Talker T
circuit_ID
ET: whatever
data

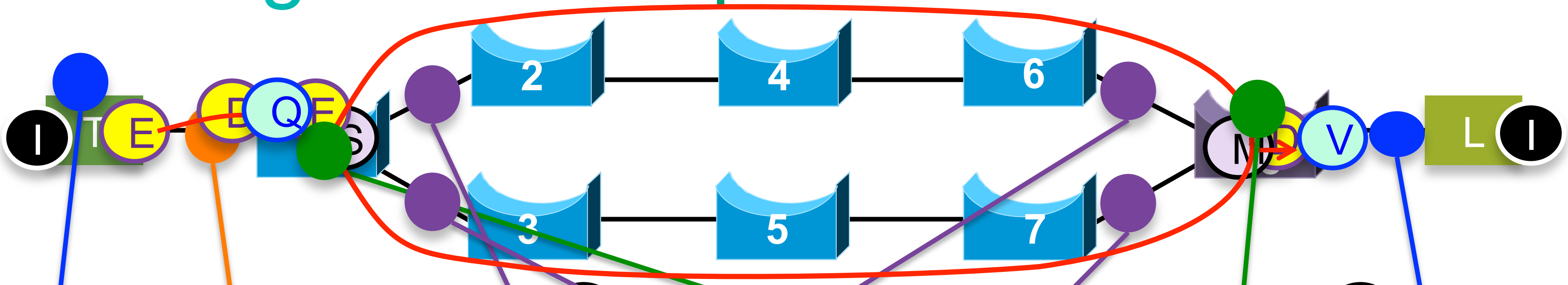
DA: TSN 734
SA: T
VLAN tag 99
ET: whatever
data

DA: TSN 734
SA: T
VLAN tag 99
ET: TSN Seq
Sequence #
ET: whatever
data

DA: Listener L
SA: Talker T
VLAN 80
ET: TSN Seq
Sequence #
ET: whatever
data

DA: Listener L
SA: Talker T
VLAN 80
ET: whatever
data

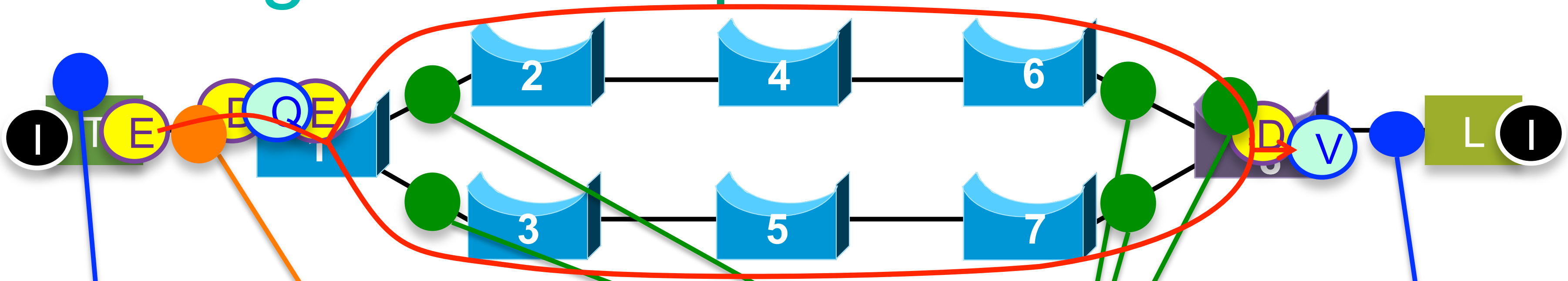
Peering relationships



- We want Split S and Merge functions M, for seamless redundancy where the **circuit** bifurcates.

		DA: TSN **7840**	DA: TSN **12**	DA: TSN 734	
		SA: T	SA: T	SA: T	
DA: Lis	DA: TSN 734	VLAN tag **23**	VLAN tag **50**	vlan_ID 99	DA: L
SA: Tal	SA: T	ET: TSN Seq	ET: TSN Seq	ET: TSN Seq	SA: T
circuit_	VLAN tag 99	Sequence #	Sequence #	Sequence #	VLAN 80
ET: wha	ET: whatever	ET: whatever	ET: whatever	ET: whatever	ET: whatever
data	data	data	data	data	data

Peering relationships



- Or, we use the normal bridge relay function, instead of explicit Split and Merge functions, and simplify things.

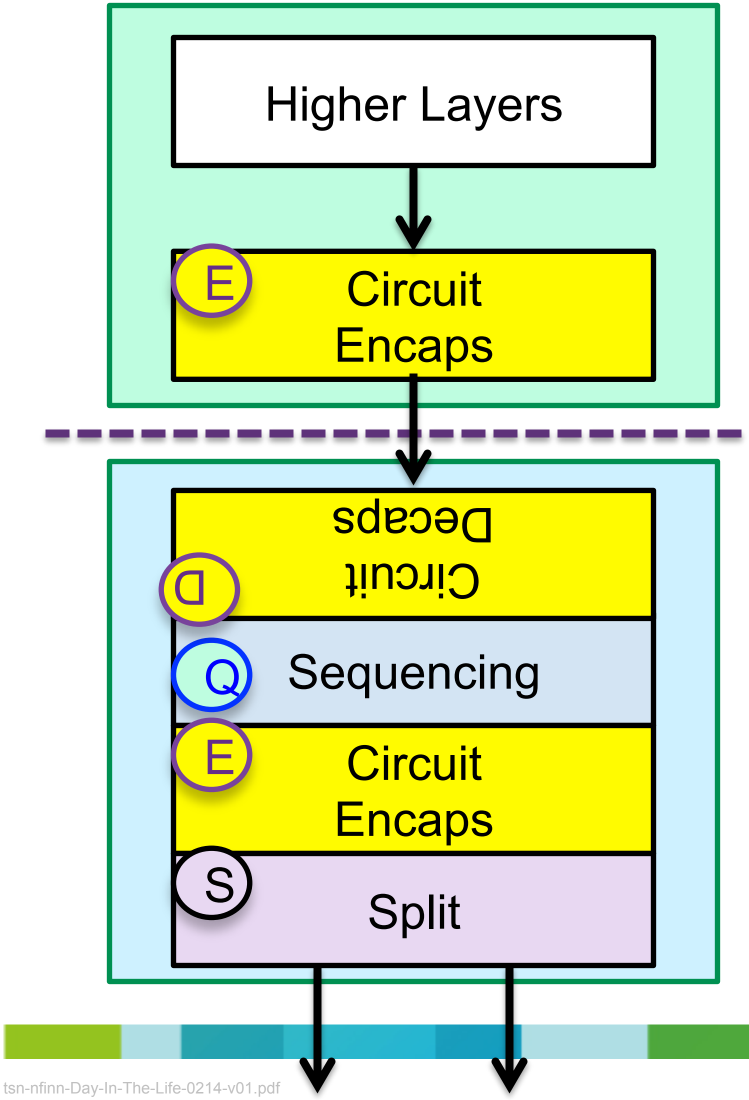
DA: Listener L
SA: Talker T
circuit_ID
ET: whatever
data

DA: TSN 734
SA: T
VLAN tag 99
ET: whatever
data

DA: TSN 734
SA: T
VLAN tag 99
ET: TSN Seq
Sequence #
ET: whatever
data

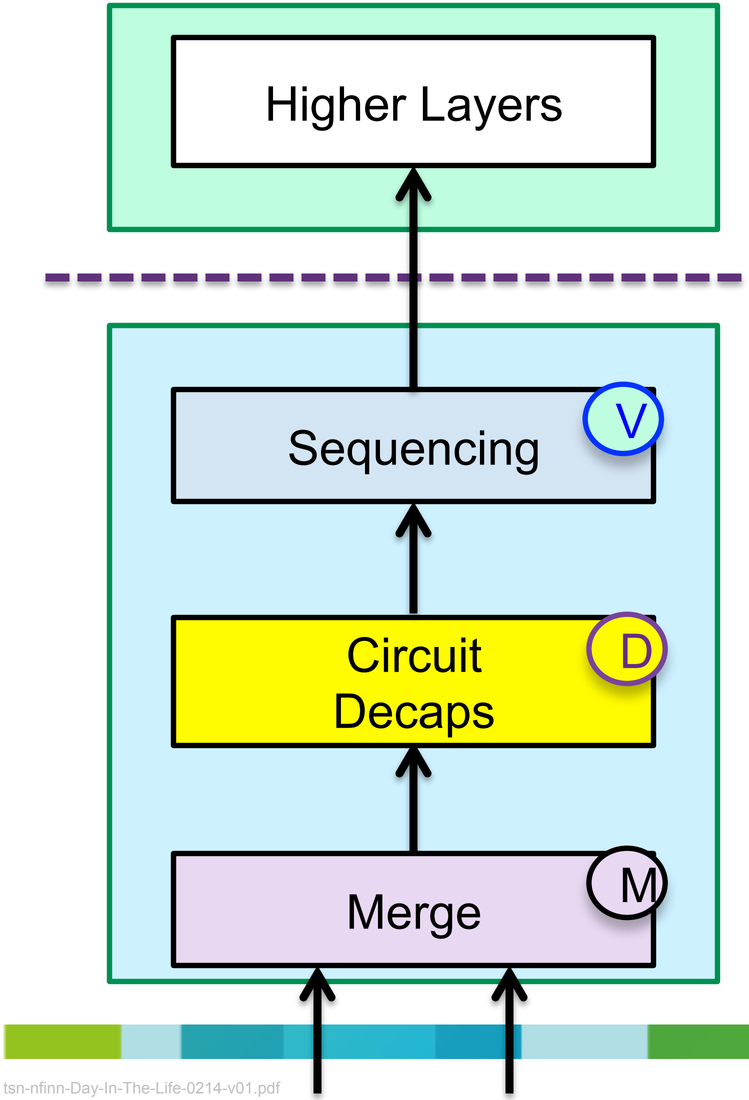
DA: L
SA: T
VLAN 80
ET: whatever
data

Peering relationships: Talker side



- In this example, the Circuit Encaps is in the **Talker** system (above the link).
- And the Sequencing is in Bridge 1 (below the link).

Peering relationships: Listener side

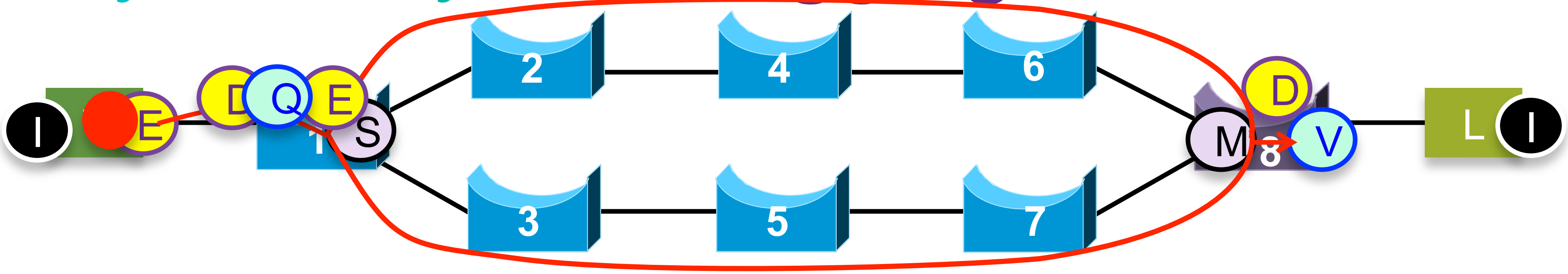


- In this example, the **Listener** system is TSN-unaware.
- And the Sequencing and TSN Decaps are both in Bridge 1 (below the link).

Sequenced TSN tagging

- Serial view

Layer 2 only: **TSN tagging**



DA: L

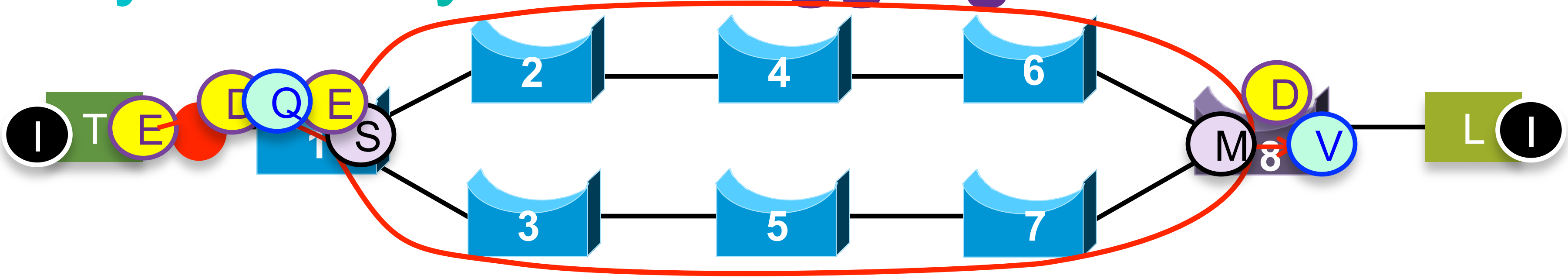
SA: T

ET: IP

IPgram

- Talker's stack is not VLAN-aware. This is what the frame is when it hits the TSN Encaps layer.
- Note that Bridge 1 would normally add a **VLAN 80 tag** to this frame.

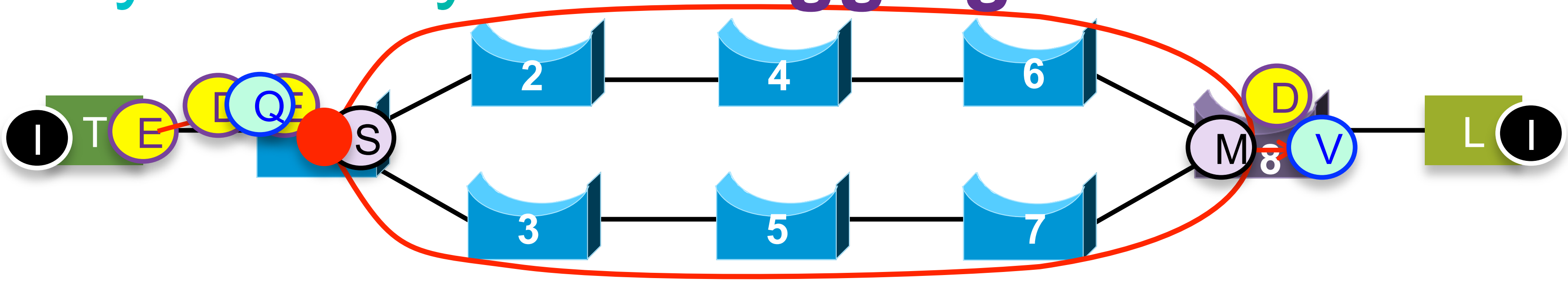
Layer 2 only: **TSN tagging**



DA: TSN 734
SA: T
VLAN tag 99
ET: IP
IPgram

- Talker is TSN-aware, so the TSN Encaps layer E adds a VLAN tag, even though Talker’s stack is not VLAN-aware.
- Talker could add sequence number, but doesn’t.

Layer 2 only: **TSN tagging**



DA: TSN 734

SA: T

vlan_identifier 99

ET: TSN Seq

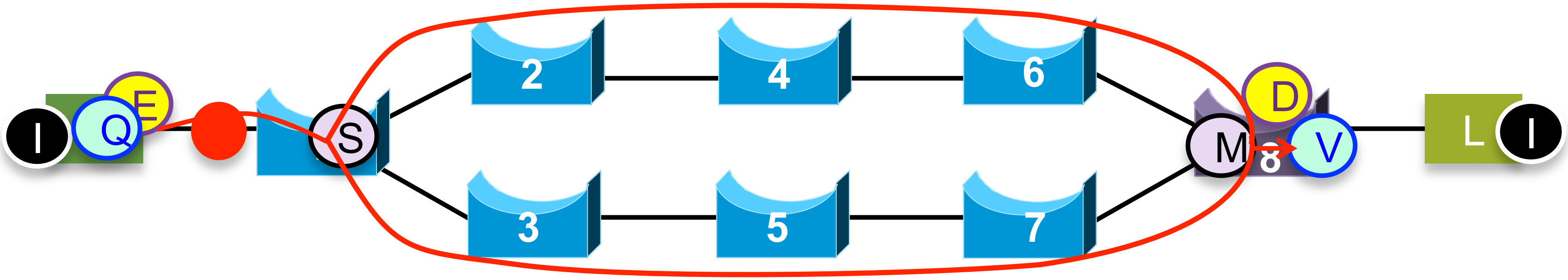
Sequence #

ET: whatever

data

- The Sequencing function Q adds a new TSN sequence number tag, to be defined by IEEE 802.1.

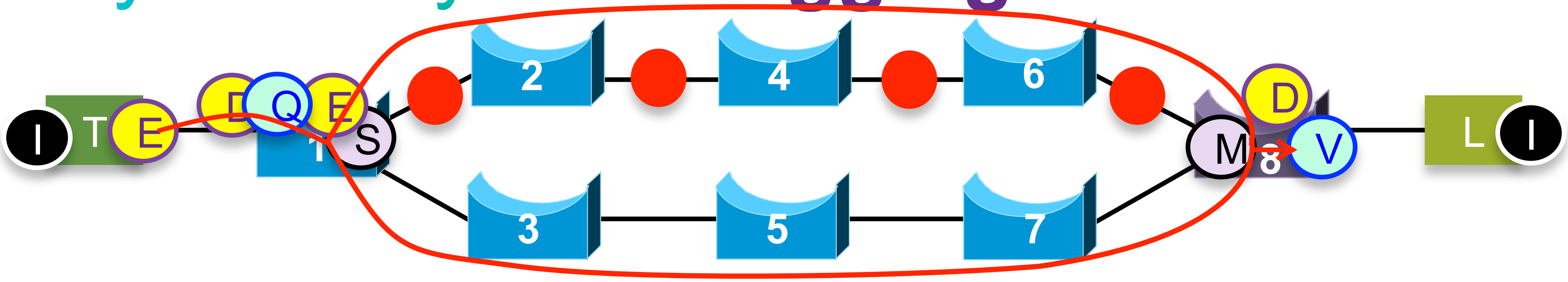
Alternative 1



DA: TSN 734
SA: T
VLAN tag 99
ET: TSN Seq
Sequence #
ET: whatever
data

- If we put the Sequencing function Q in the host, then we have a better capability to detect packet losses.
- Also, packets can be more easily buffered for in-order delivery.

Layer 2 only: **TSN tagging**

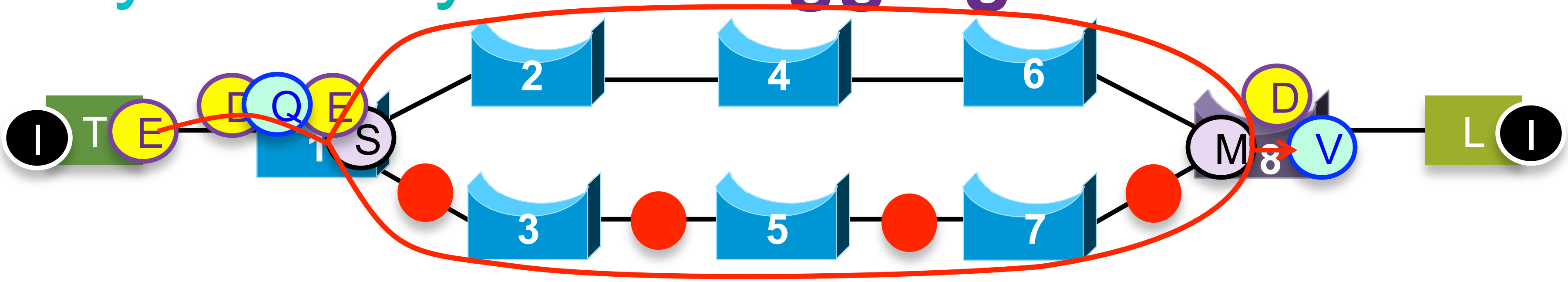


●

DA: TSN 7840
SA: T
VLAN tag 23
ET: TSN Seq
Sequence #
ET: whatever
data

- The Split function Ⓢ replicates the packet on two interfaces.
- It takes TSN 734 in, and splits it into TSN 7840 (upper link) and TSN 12 (lower).

Layer 2 only: **TSN tagging**

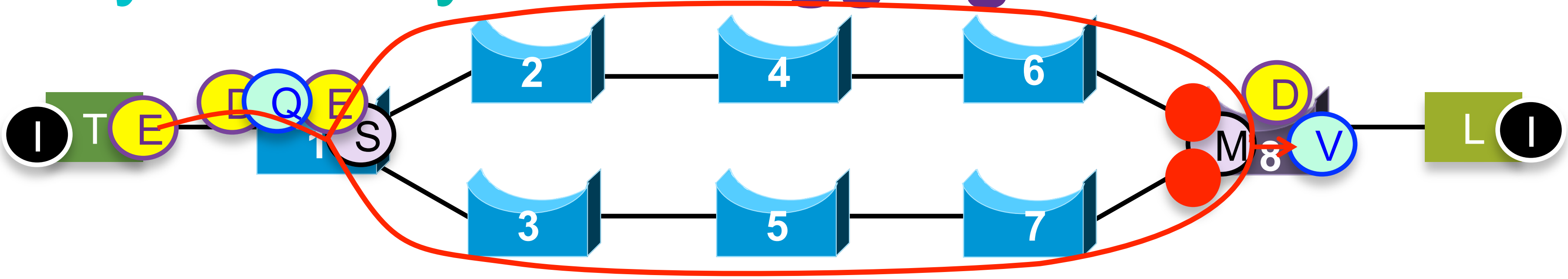


●

DA: TSN 12
SA: T
VLAN tag 50
ET: TSN Seq
Sequence #
ET: whatever
data

- Note that we have a different circuit ID on the second path.
- Another presentation is required to discuss whether the DA, the VLAN, both, or neither, should be different.

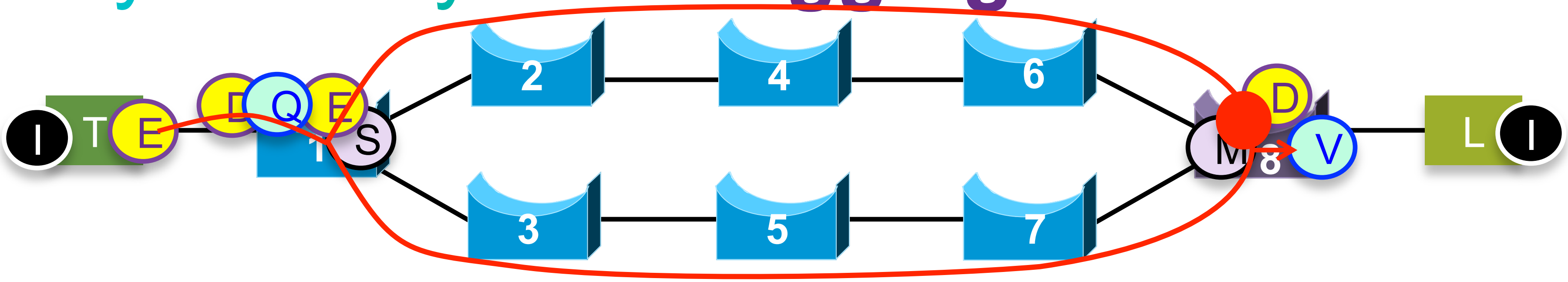
Layer 2 only: **TSN tagging**



DA: TSN 7840 or 12
SA: T
VLAN tag 23 or 50
ET: TSN Seq
Sequence #
ET: whatever
data

- The Merge function has to operate on circuit ID (MAC DA + VLAN) and sequence number (in TSN Sequence tag).

Layer 2 only: **TSN tagging**



DA: TSN 734

SA: T

vlan_identifier 99

ET: TSN Seq

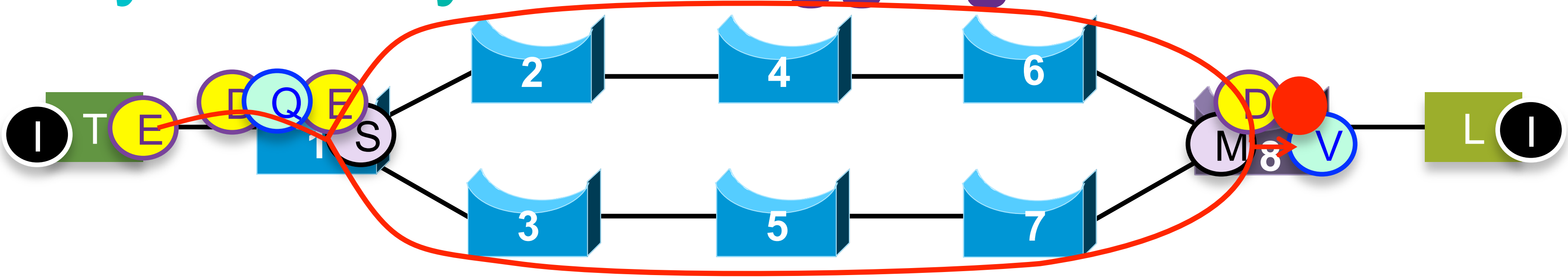
Sequence #

ET: whatever

data

- Output from Merge function Ⓜ
- Note that TSN 7840[23] and TSN 12[50] were combined into TSN 734[99], the original path from Talker T.

Layer 2 only: **TSN tagging**



DA: L

SA: T

circuit_identifier

ET: TSN Seq

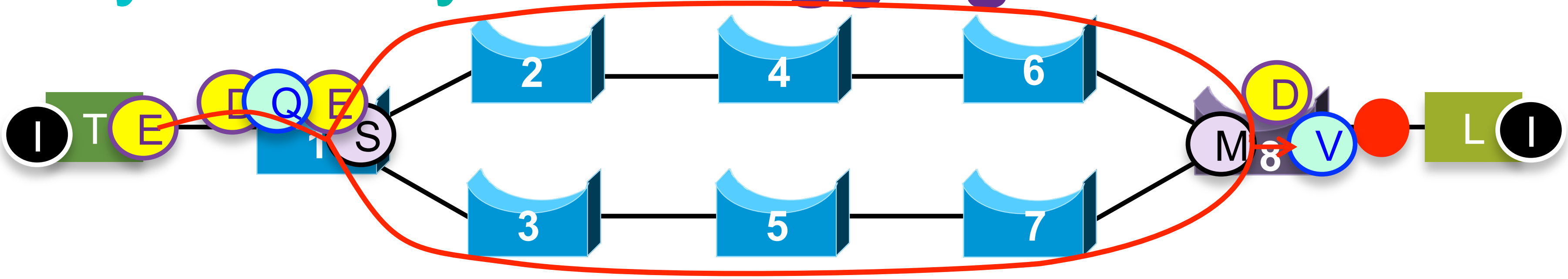
Sequence #

ET: whatever

data

- The TSN Decapsfunction D then removes the sequence number.
- (The circuit_identifier and vlan_identifier are still present as parameters.)

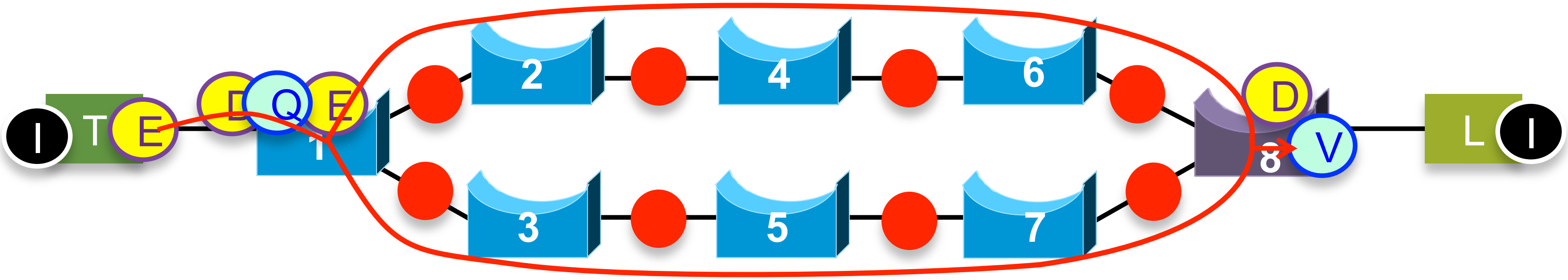
Layer 2 only: **TSN tagging**



DA: L
SA: T
VLAN tag 80
ET: whatever
data

- Output from Sequencing function V is what would have been output from the Talker, modulo the VLAN tag changes the bridges would make.
- (This knowledge came to Bridge 8 via the control protocol.)

Alternative 2

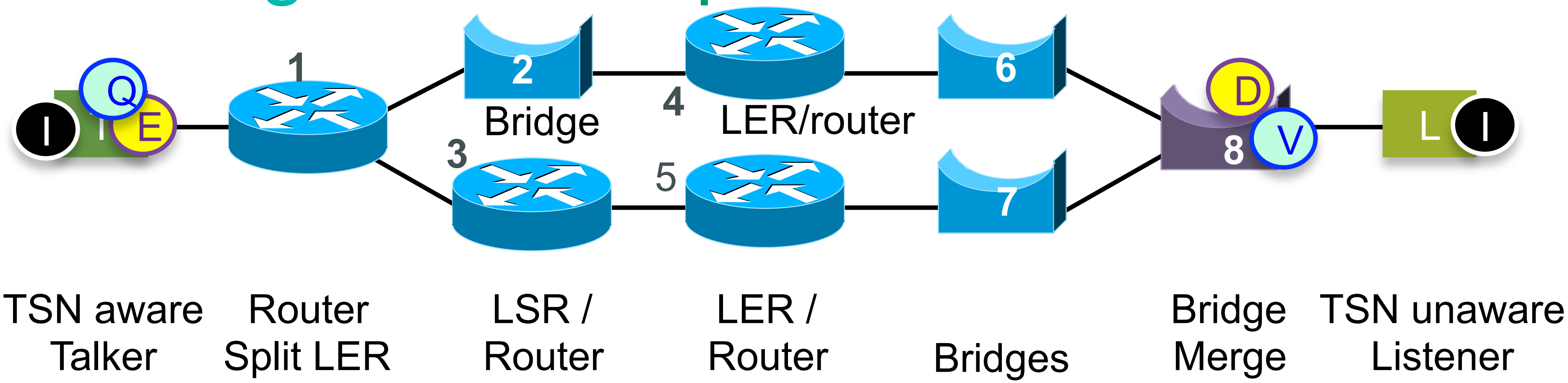


DA: TSN 734
SA: T
VLAN tag 99
ET: TSN Seq
Sequence #
ET: whatever
data

- Or, we eliminate the Split Ⓢ and Merge Ⓜ functions.
- We use TSN 734[99] everywhere.

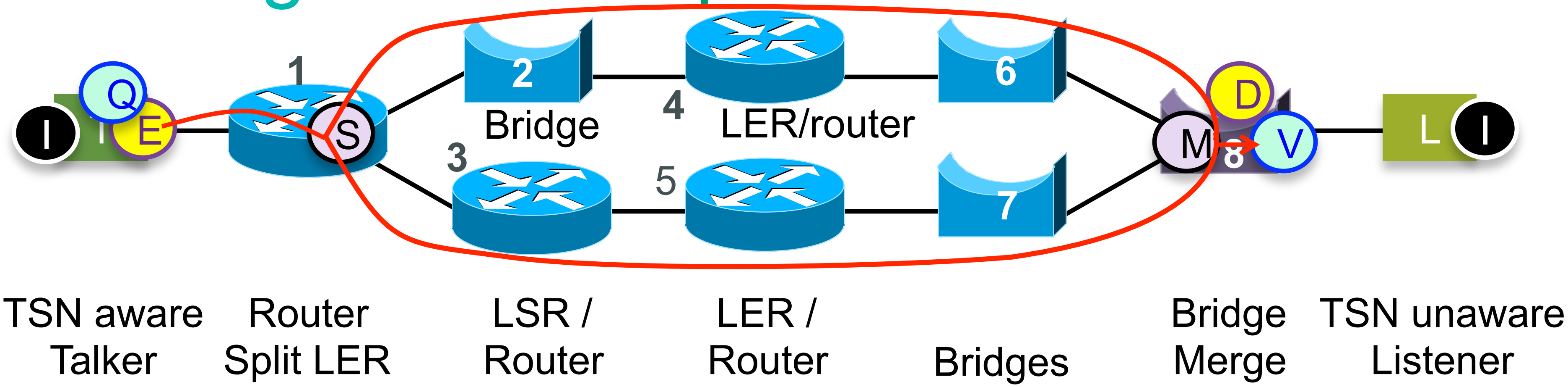
Case 2:
Mixed L2/L3 using IPgram pseudowires and Sequenced TSN

Peering relationships



- A single-port TSN-aware, VLAN-aware Talker and a single-port TSN-unaware, VLAN-unaware Listener.
- Talker attached to a router; Listener to a bridge.
- A network consisting of a variety of routers and bridges.

Peering relationships



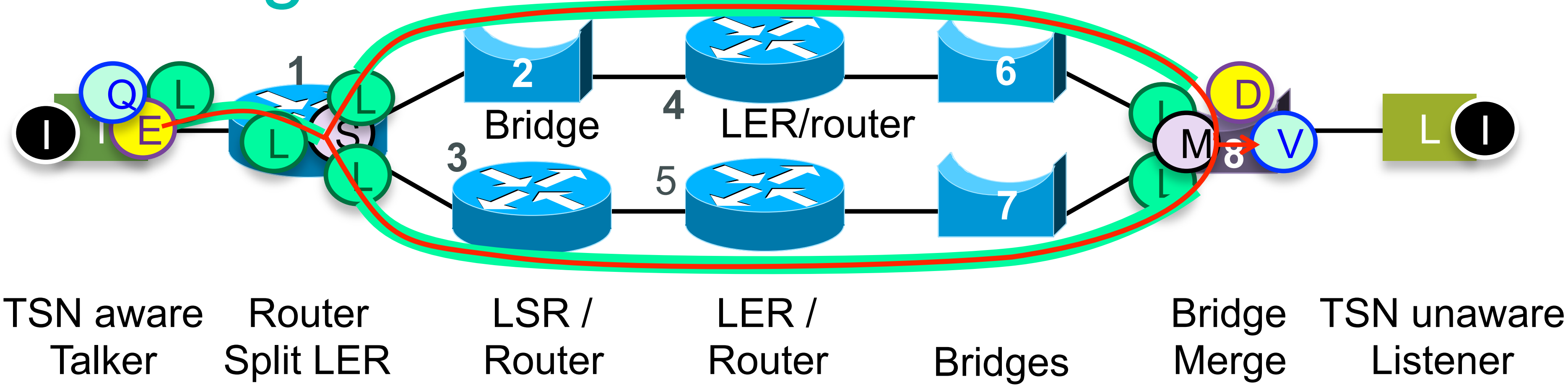
Ⓢ Ⓜ Router 1 and Bridge 8 are the **split/merge (seamless redundancy)** peers, because they split and merge the circuits.

- They operate on the **circuit**.

Now, things start to get complicated

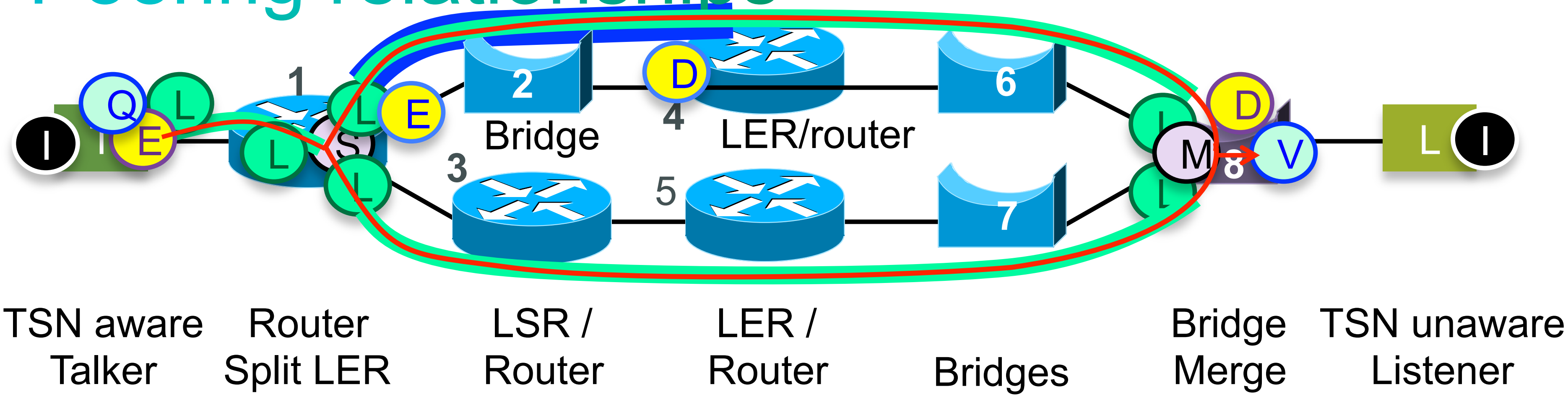
- We're going to build this example up with the peering relationships as they are perceived by the Talker, then Router 1, and so on, through the network.
- As we proceed we will modify these perceptions, until we see the whole picture.
- This seems the easiest way to understand the data flow. It is not necessarily the easiest way to understand the control flows.

Peering relationships



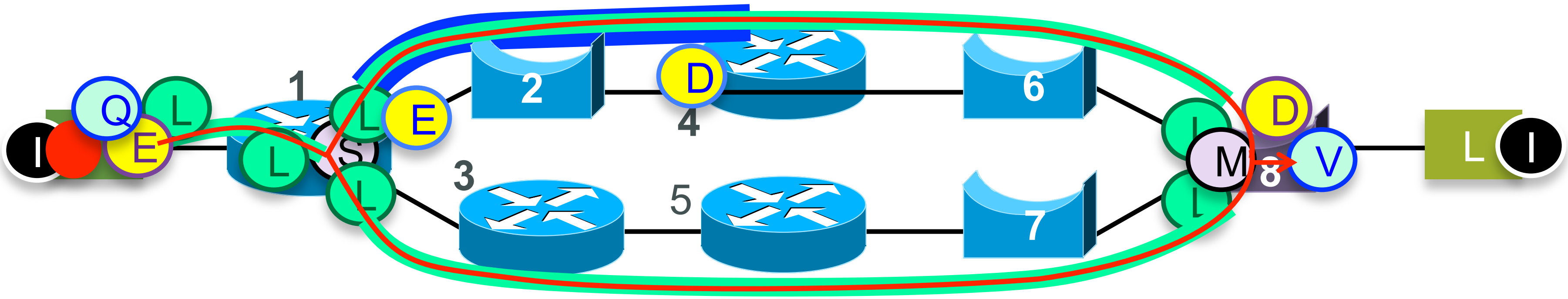
- A network of **Label Switched Paths (LSPs)** connects E to S to M to D. Each endpoint is a Label Edge Router (LER) function.
- The fixed paths are not integral to seamless redundancy; they carry the circuit to the splitter and merger.

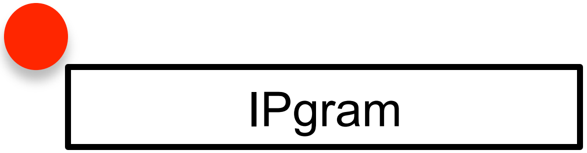
Peering relationships



E D An extra TSN Circuit Encaps/Decaps pair is needed to convey the circuit over the Bridged LAN represented by Bridge 2.

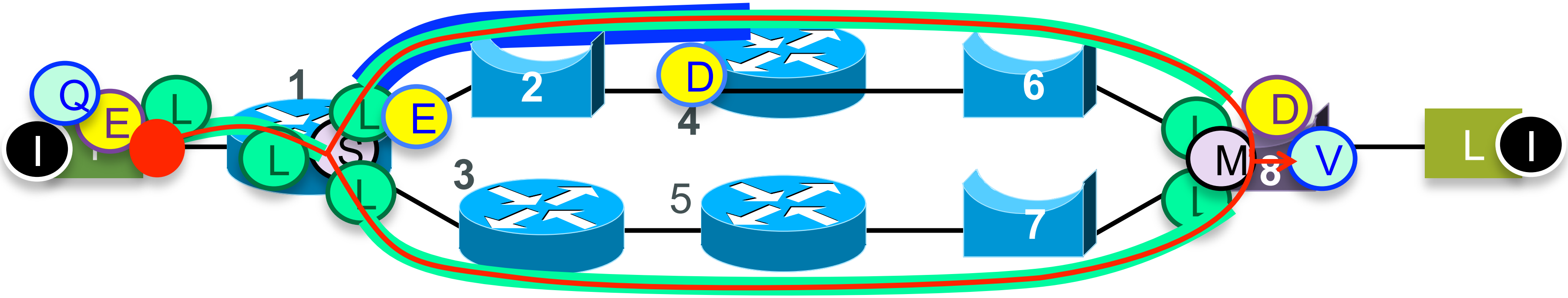
A day in the life of a packet





- Talker T has an IPgram to send to Listener L.

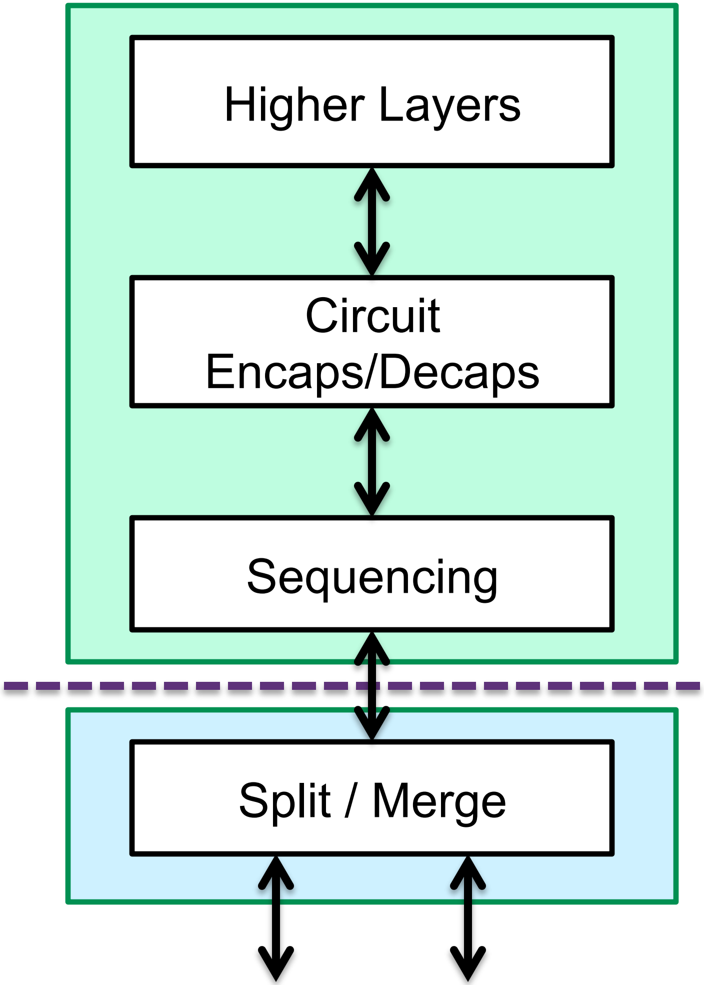
A day in the life of a packet



pseudowire label 28
control (sequence)
IPgram

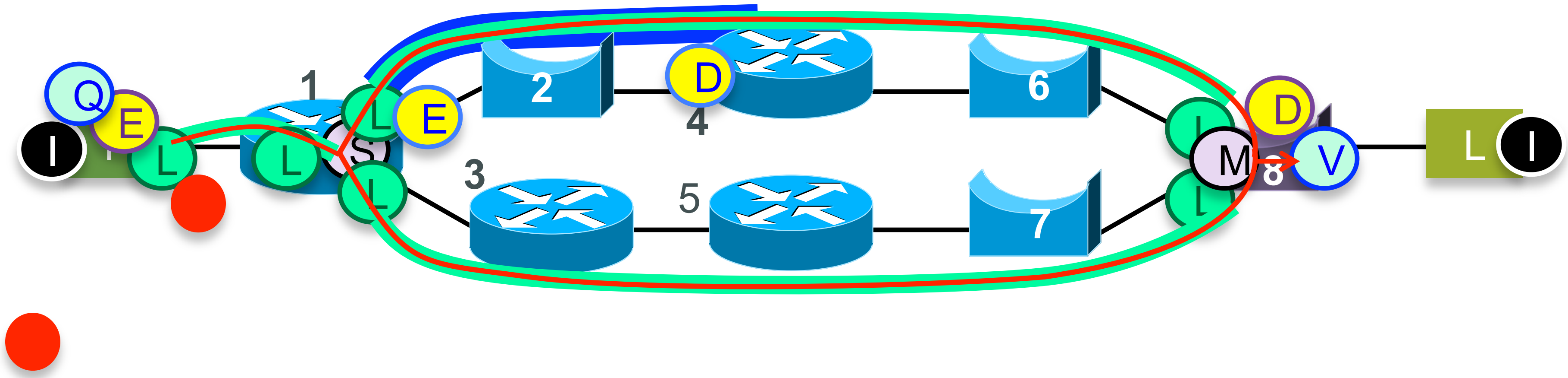
- Talker T’s combined TSN Encaps E and Sequencing Q functions use an **IPgram pseudowire** for the circuit.
- Bridge 8’s functions D V are at the other end of the network.

A day in the life of a packet



- Note that this is the layering – the top box is Talker T, and the bottom box is Router 1.
- Note that the sequence number can be used (at the far end) to detect packet loss between Talker T and Router 1.

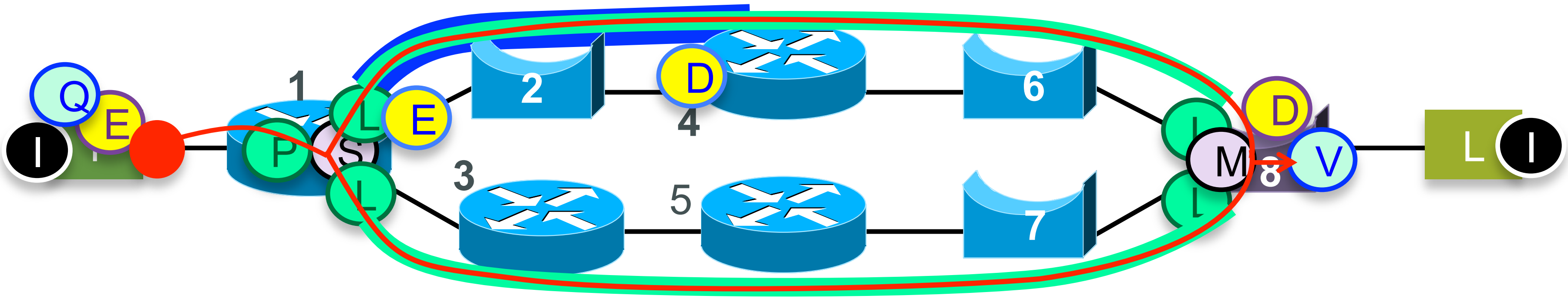
A day in the life of a packet



label 60
pseudowire label 28
control (sequence)
IPgram

- In the general case, the LER function L would encapsulate the pseudowire would be carried in an LSP.

A day in the life of a packet



~~label 60~~

pseudowire label 28

control (sequence)

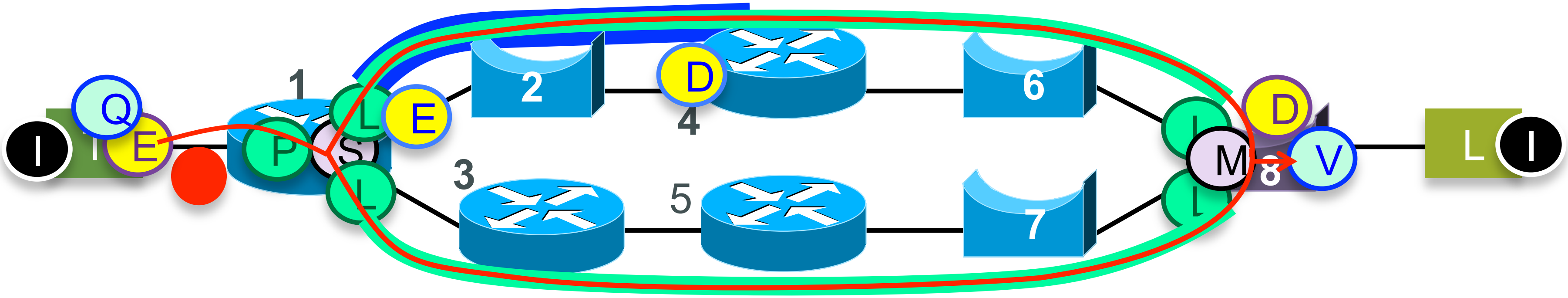
IPgram

- In this particular case, we will assume that Router 1 is doing a “Penultimate Hop P Pop” (PHP) function. That eliminates the need for the outside label.

Warning

- The PHP step may be controversial.
- Perhaps there is another MPLS label, a path label, on the frame between the Talker and Router 1.

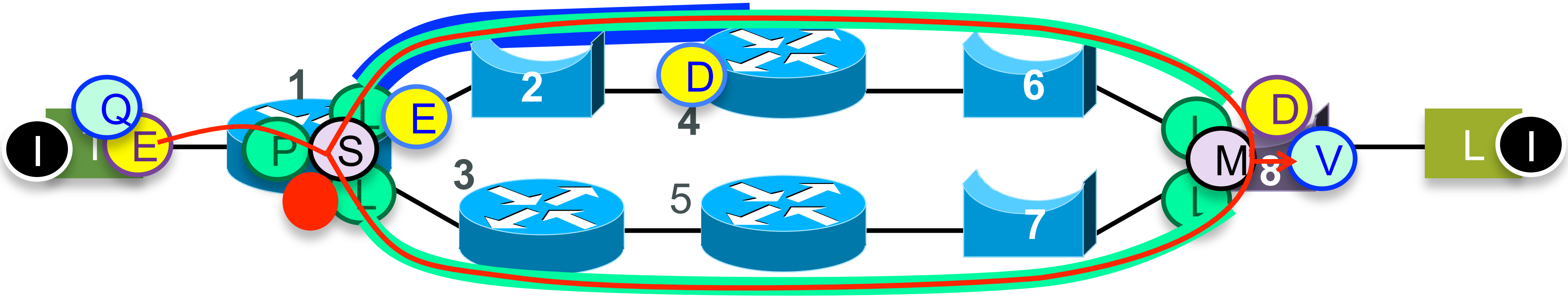
A day in the life of a packet



DA: Router 1
SA: T
ET: MPLS
pseudowire label 28
control (sequence)
IPgram

- So, the frame from Talker T to Router 1 looks like this on the Ethernet between Talker T and Router 1.

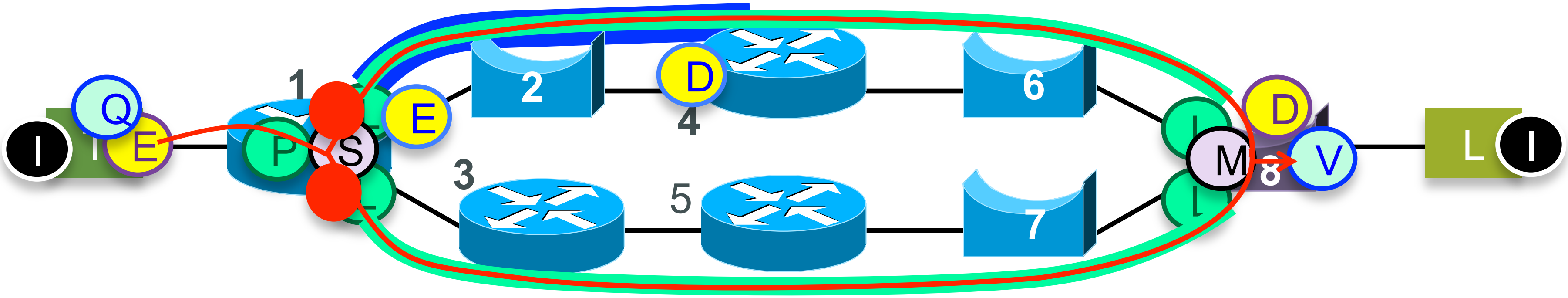
A day in the life of a packet



pseudowire label 28
control (sequence)
IPgram

- The Splitter function S in Router 1 replicates the pseudowire and inserts into two LSPs, one using the upper path, and one using the lower path.

A day in the life of a packet



●

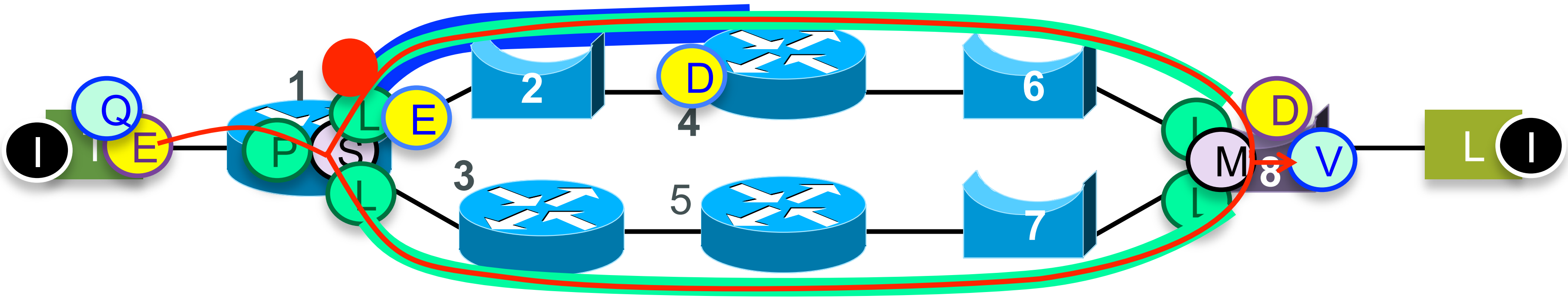
pseudowire label 419
control (sequence)
IPgram

●

pseudowire label 31
control (sequence)
IPgram

- The Splitter function Ⓢ has split the one pseudowire 28 into two pseudowires 419 and 31, copying the one control word to both of them.

A day in the life of a packet



DA: Router 4

SA: Router 1

vlan_identifier 15

ET: MPLS

Tunnel label 51

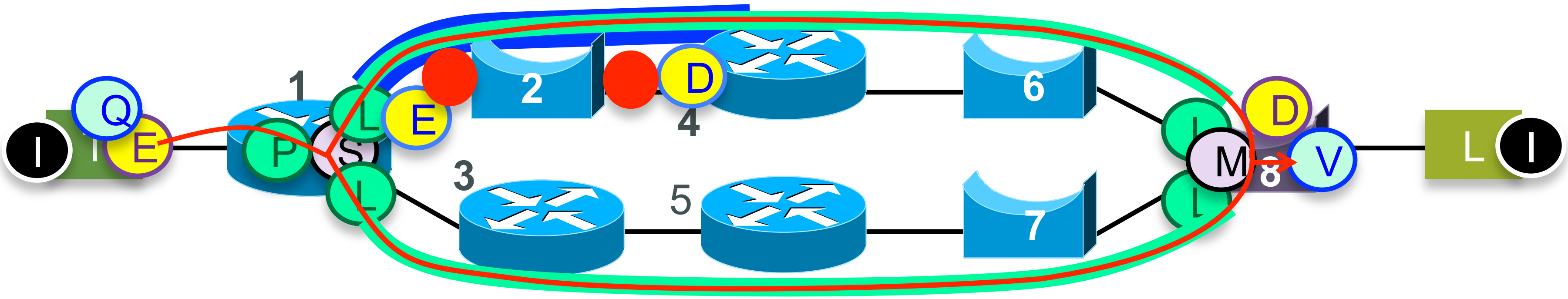
pseudowire label 419

control (sequence)

IPgram

- The upper tunnel **would have** look like this, on the wire, when labeled with Tunnel 51, **except that** …

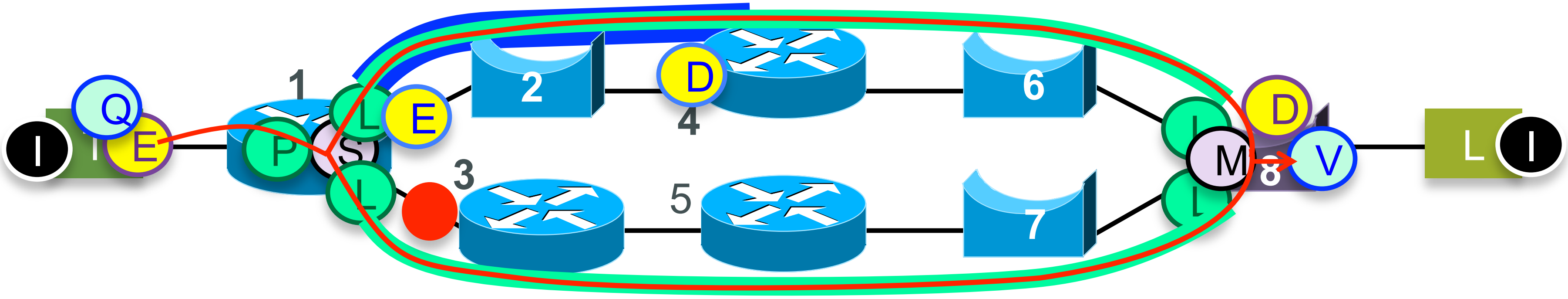
A day in the life of a packet



DA: TSN 140
SA: Router 1
VLAN tag 309
ET: MPLS
Tunnel label 419
pseudowire label 28
control (sequence)
IPgram

- Router 1 and Router 4 are separated by a TSN bridged network, so require a TSN encapsulation D E .
- This gets the packet to Router 4.

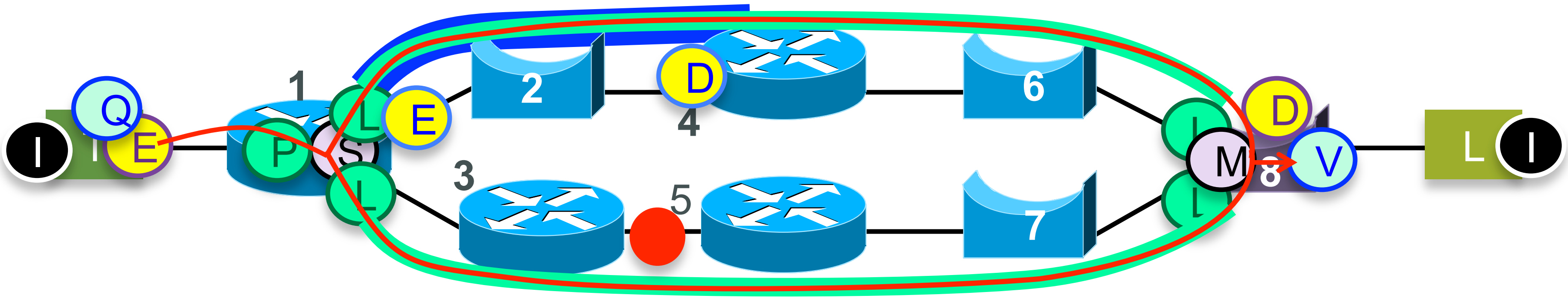
A day in the life of a packet



DA: Router 3
SA: Router 1
ET: MPLS
Tunnel label 557
pseudowire label 31
control (sequence)
IPgram

- Meanwhile, Router/LER 1, Router/LSR 3 and Router/LSR 5 are moving the LSP packet along.

A day in the life of a packet

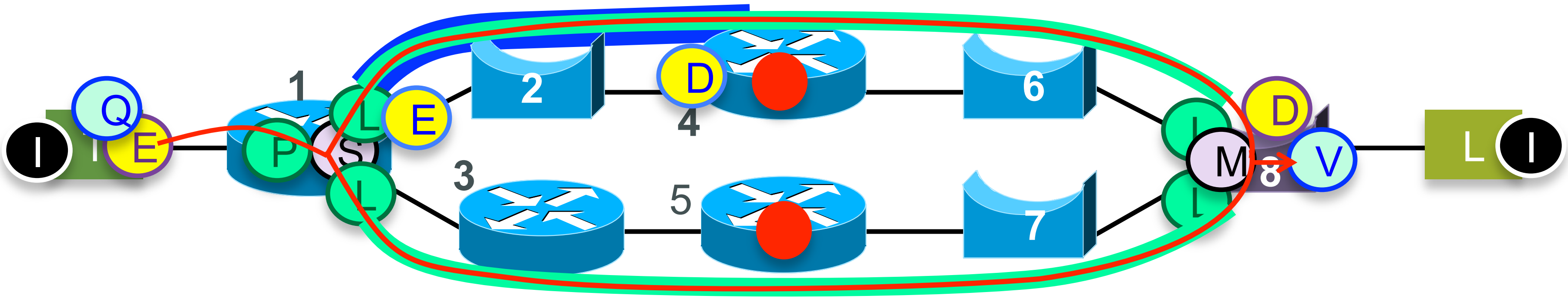


●

DA: Router 5
SA: Router 3
ET: MPLS
Tunnel label 346
pseudowire label 31
control (sequence)
IPgram

- Meanwhile, Router/LER 1, Router/LSR 3 and Router/LSR 5 are moving the LSP packet along.
- Router/LSR 3 changes the Tunnel label 557→346.

A day in the life of a packet



●

Tunnel label 51
pseudowire label 419
control (sequence)
IPgram

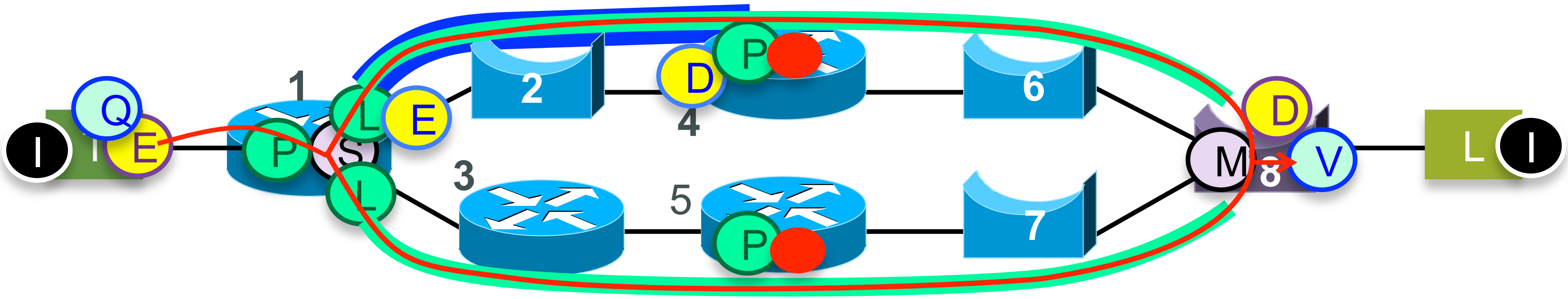
- Router 4 now has this labeled packet.

●

Tunnel label 346
pseudowire label 31
control (sequence)
IPgram

- And Router 5 has this one.

A day in the life of a packet



●

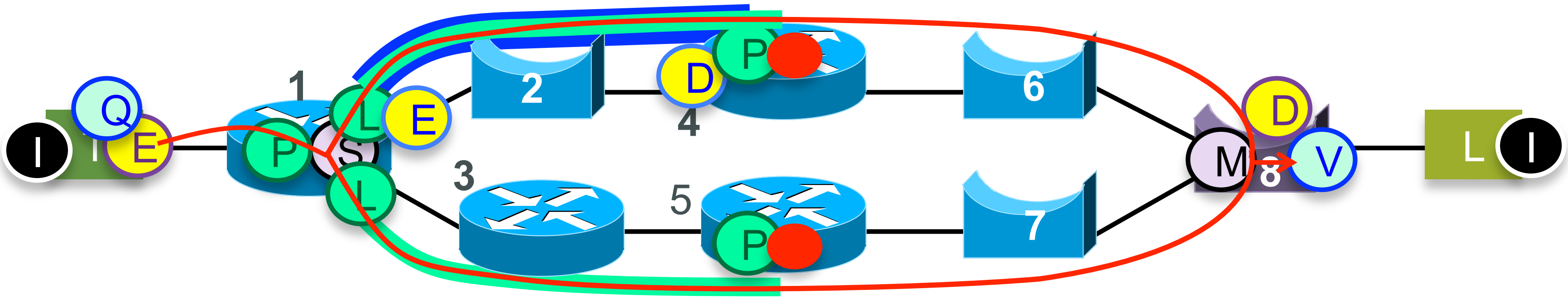
pseudowire label 419
control (sequence)
IPgram

●

pseudowire label 31
control (sequence)
IPgram

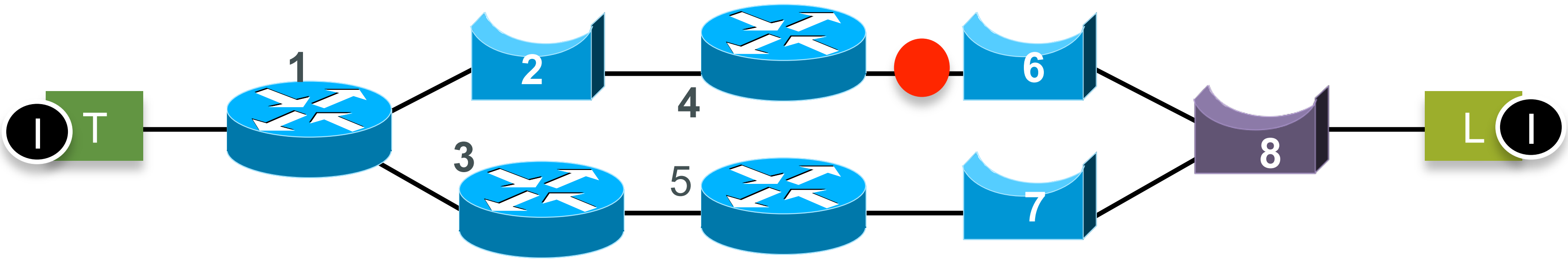
- For the sake of reduced frame size, Router/LSPs 4 and 5 perform PHP, Ⓟ which eliminates Tunnel labels 51 and 346 (and the LERs Ⓛ in Bridge 8).

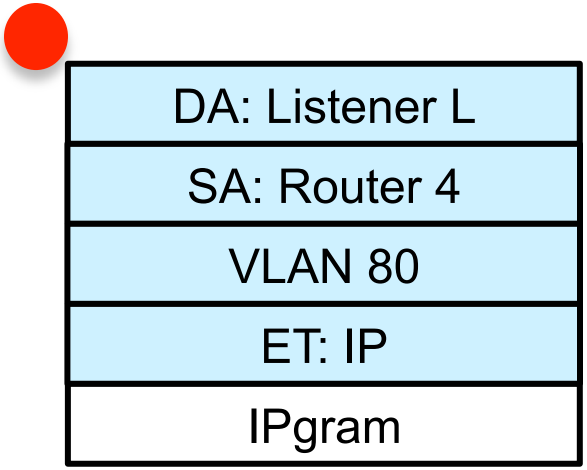
A day in the life of a packet



- One can argue the semantics of the green tunnels. In theory, each tunnel continues to its natural end at Bridge 8. The control plane may maintain this. But, in the data plane, the tunnel label disappears.
- So, we will shorten the tunnel in the diagram to match the data plane encapsulation

Interlude: the Interworking function





DA: Listener L

SA: Router 4

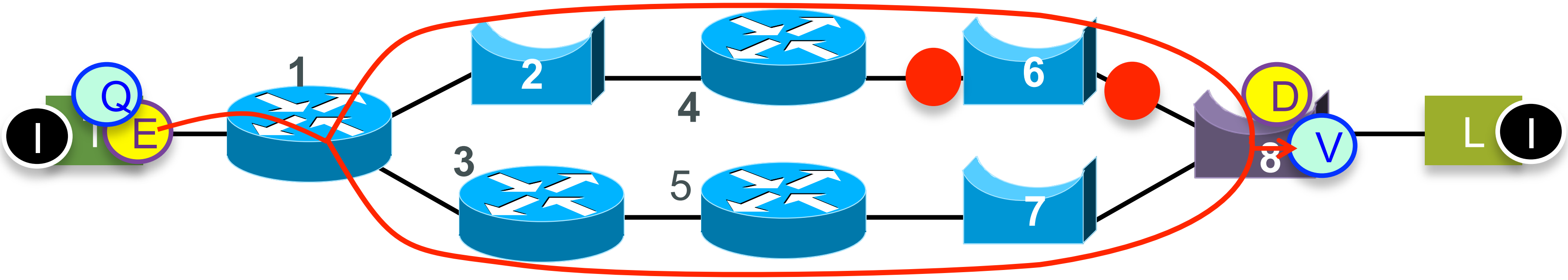
VLAN 80

ET: IP

IPgram

- Without all this tunneling, Router 4 would normally (see above) add this Ethernet encapsulation to the original IPgram in order to get it to its destination through the right-hand bridged network.

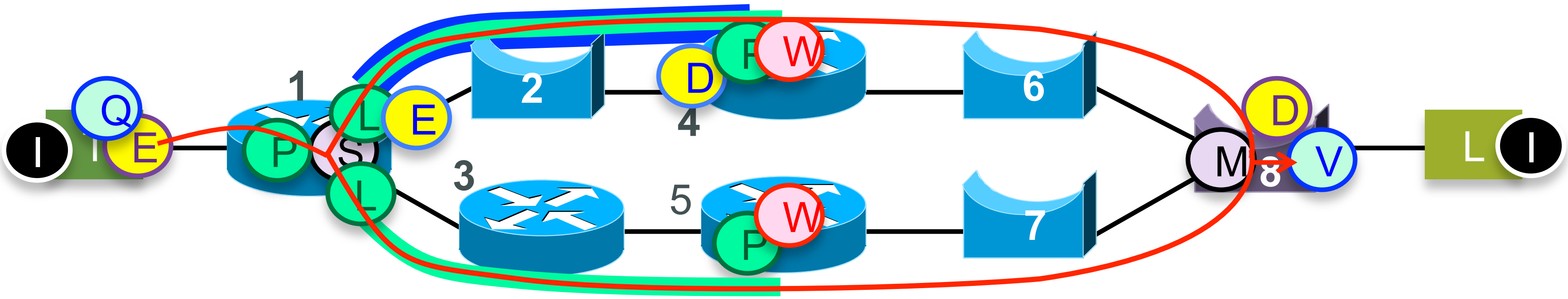
Interlude: the Interworking function



DA: TSN 7840
SA: Router 4
VLAN tag 23
ET: IP
IPgram

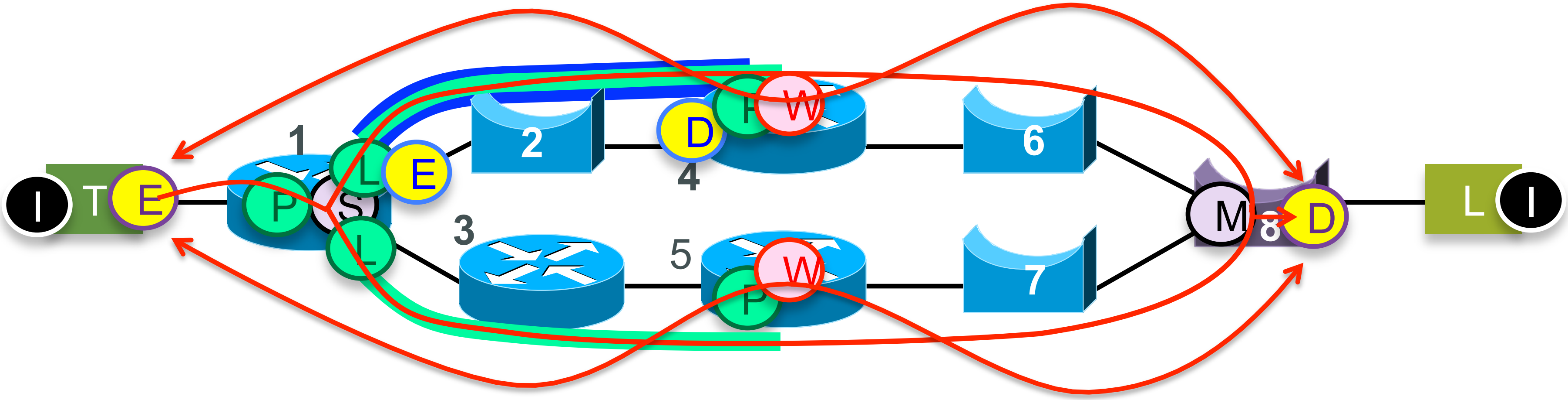
- As we have seen, the TSN Circuit Encaps function E and Sequencing function Q turn that frame into this format.

Interlude: the Interworking function



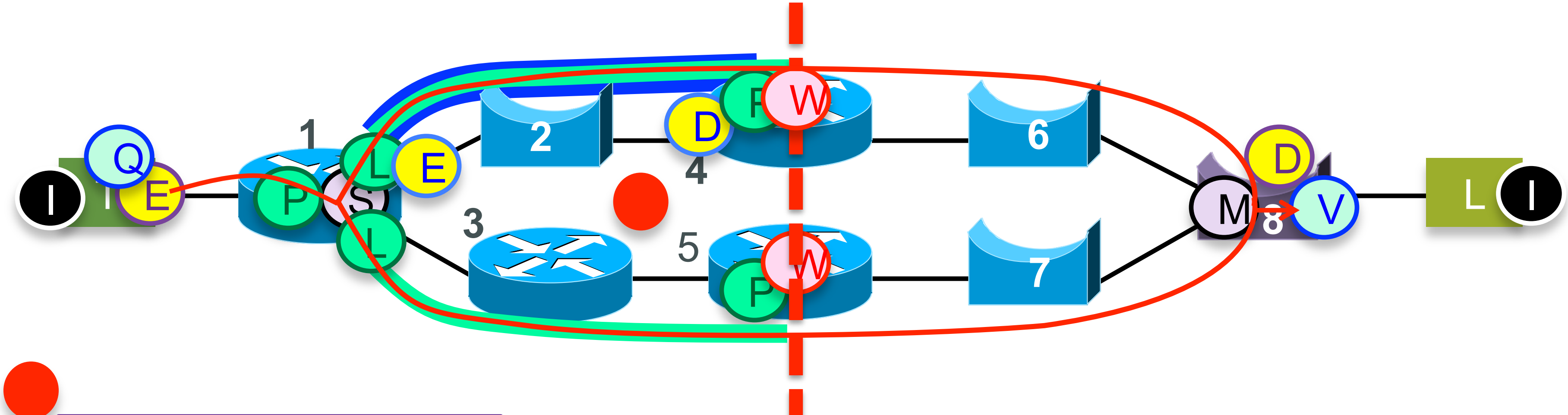
- So, we introduce an Interworking Function W
- The Interworking Function transports the Serialization layer across a layering gap caused by a change in Circuit Encaps/ Decaps functions.

Interlude: the Interworking function



- The Interworking Functions W enable the TSN Circuit Encaps function E and the Decaps function D at the very ends of the network to be **peers**, just like the Ethernet end-to-end case.

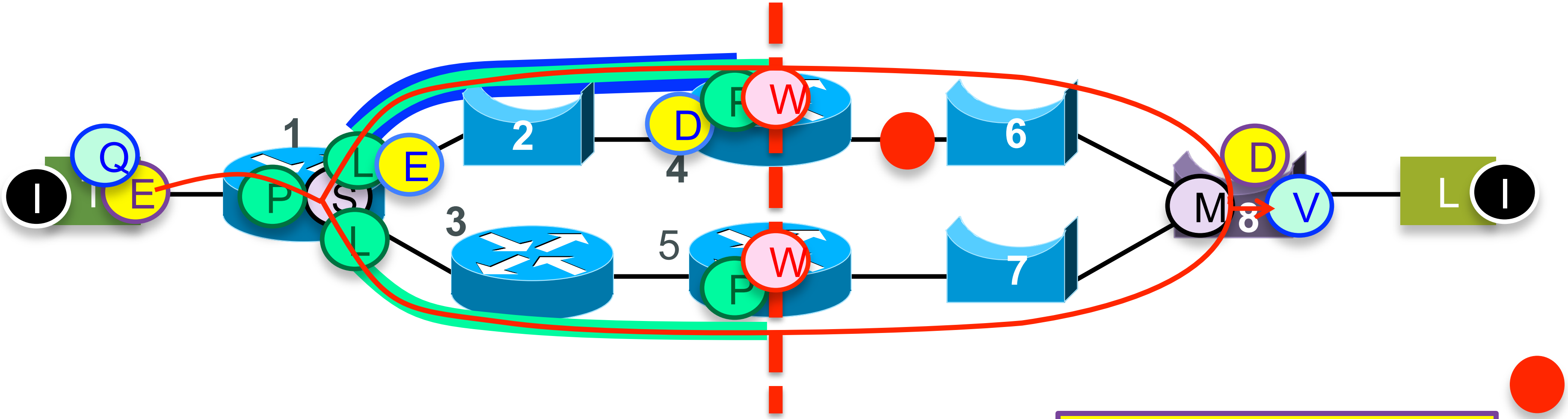
Interlude: the Interworking function



pseudowire label 419
control (sequence)
IPgram

- In the left-hand world, the Circuit ID Encaps/Decaps is an **IPgram pseudowire**.

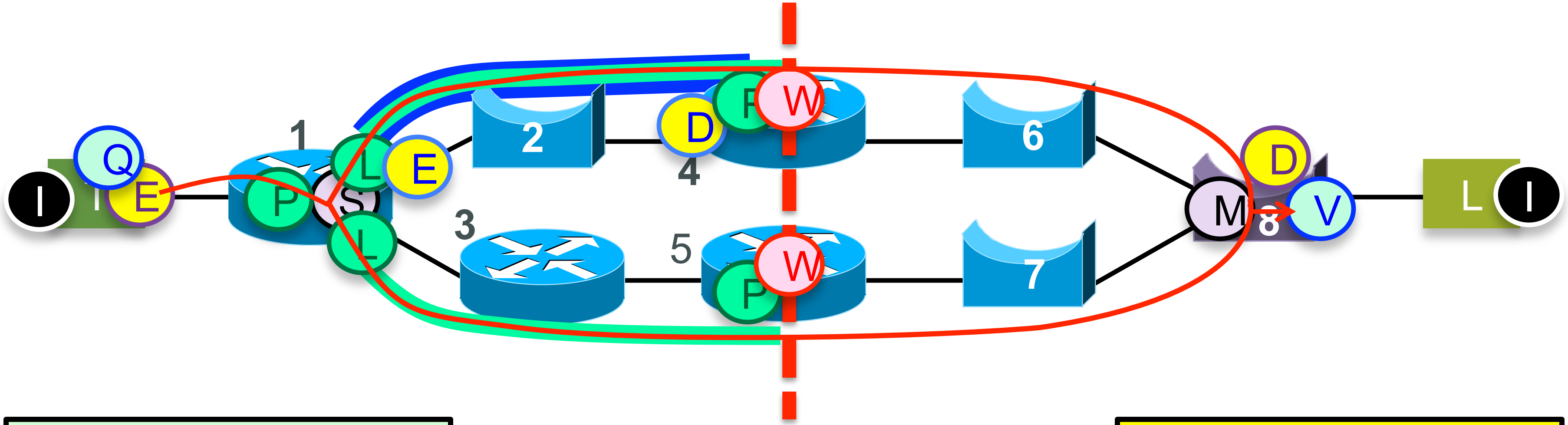
Interlude: the Interworking function



- In the right-hand world, the Circuit ID Encaps/Decaps is the **Serialized TSN encaps**.

DA: TSN 7840
SA: Router 4
VLAN tag 23
ET: TSN Seq
Sequence #
ET: IP
IPgram

Interlude: the Interworking function



pseudowire label 419

control (sequence)

IPgram

DA: TSN 7840

SA: Router 4

VLAN tag 23

ET: TSN Seq

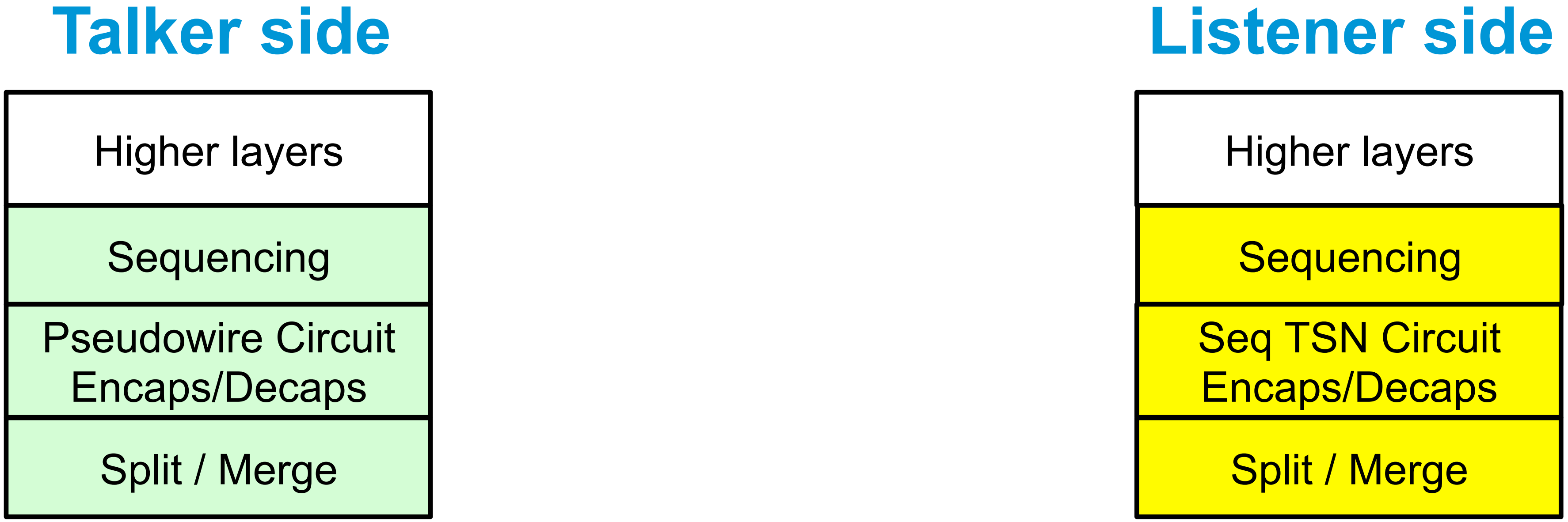
Sequence #

ET: IP

IPgram

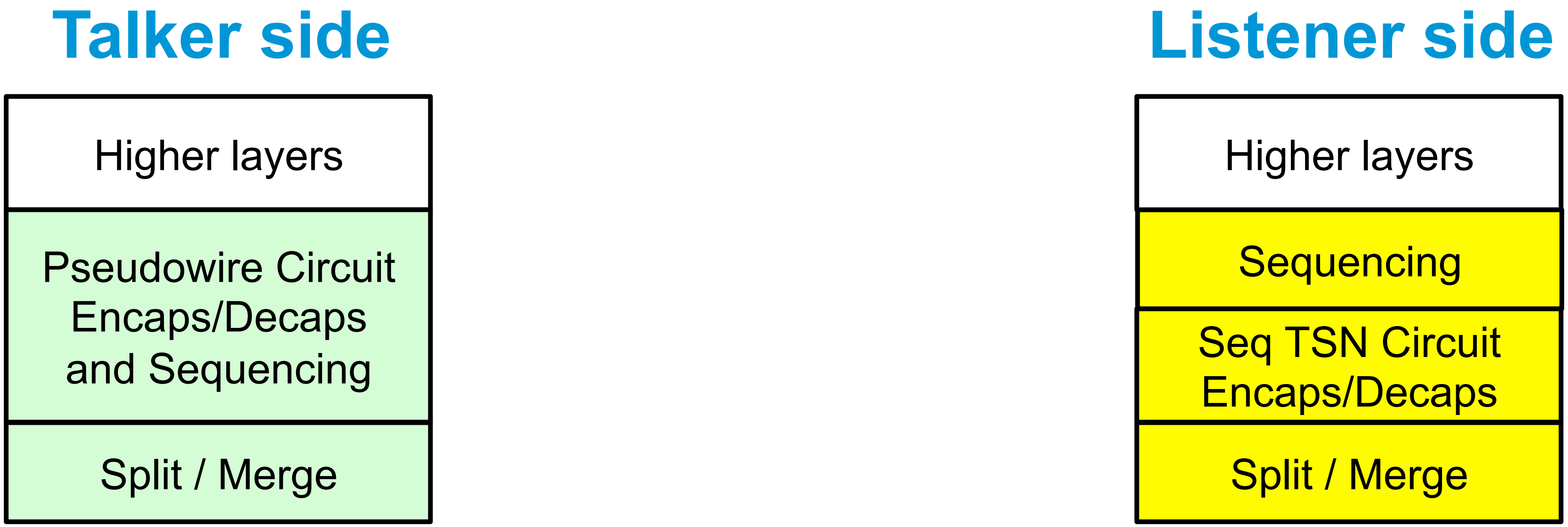
- The Interworking function W carries the **Sequence number** across the gap.

Interlude: the Interworking function



- We have two differet protocol stacks, **pseudowire** and **sequenced TSN**, that perform essentially the same function.
- We want them to peer with each other.

Interlude: the Interworking function



Talker side

Higher layers

Pseudowire Circuit Encaps/Decaps and Sequencing

Split / Merge

Listener side

Higher layers

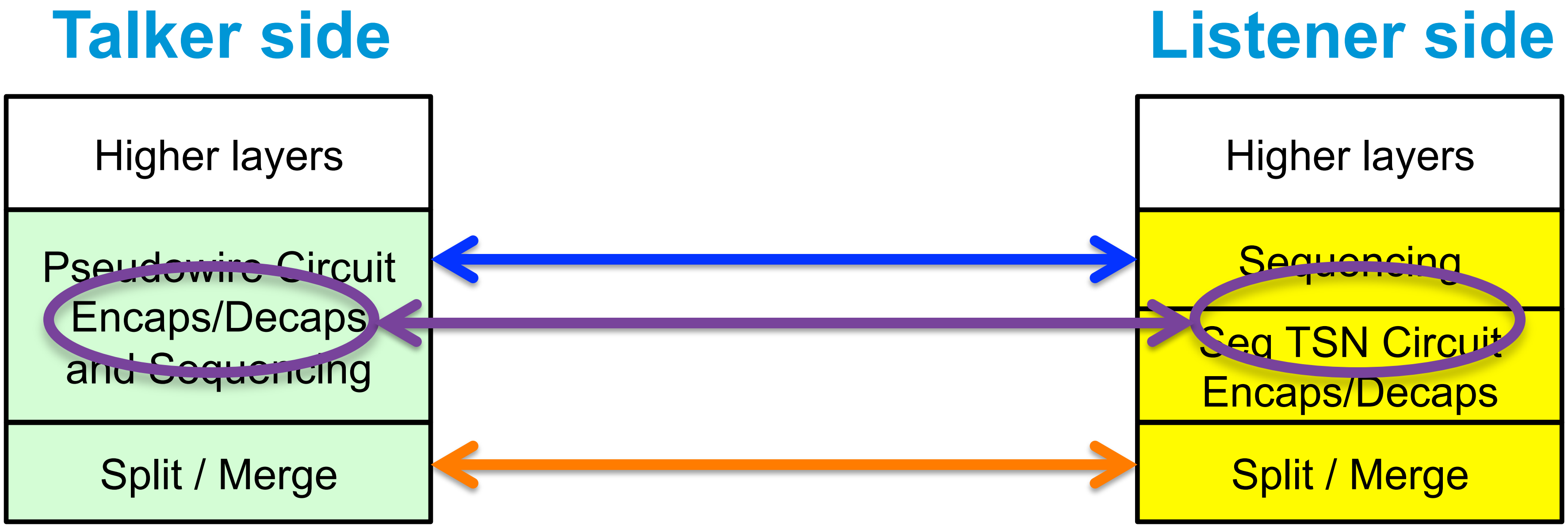
Sequencing

Seq TSN Circuit Encaps/Decaps

Split / Merge

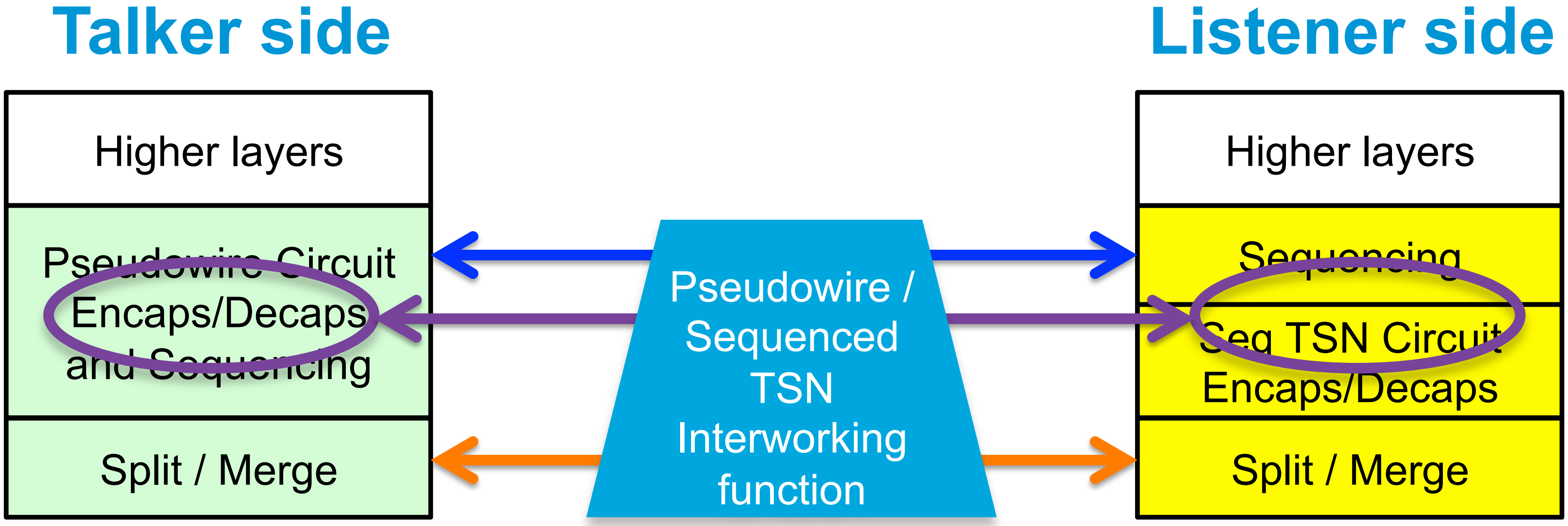
- Note that the pseudowire encapsulation and functional description includes sequencing.

Interlude: the Interworking function



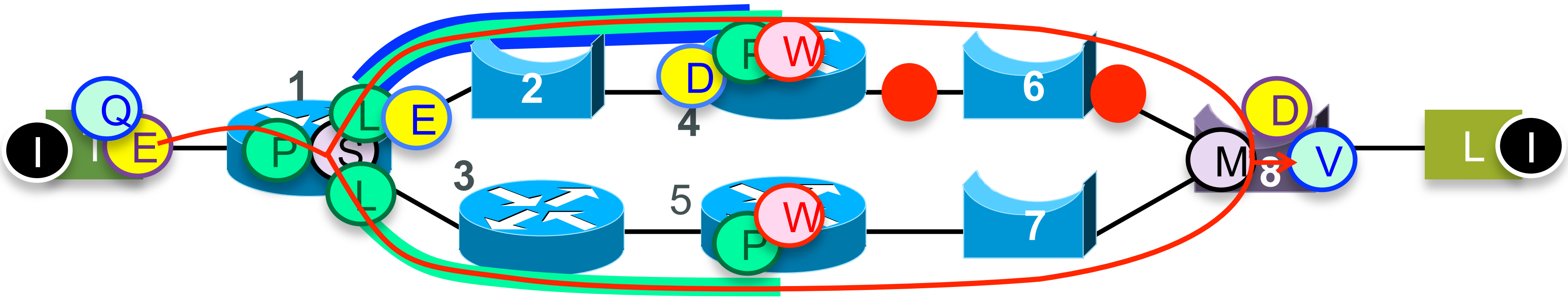
- Because the **sequence numbers** are similar (preferably, the same!), the **Split/Merge** functions have **no tag layer**, and we choose to have **no other layer** between the TSN Circuit ID and Sequencing layers, . . .

Interlude: the Interworking function


Talker side
Listener side
Higher layers
Pseudowire Circuit Encaps/Decaps and Sequencing
Split / Merge
Pseudowire / Sequenced TSN Interworking function
Higher layers
Sequencing
Seq TSN Circuit Encaps/Decaps
Split / Merge

- . . . an Interworking function can succeed.

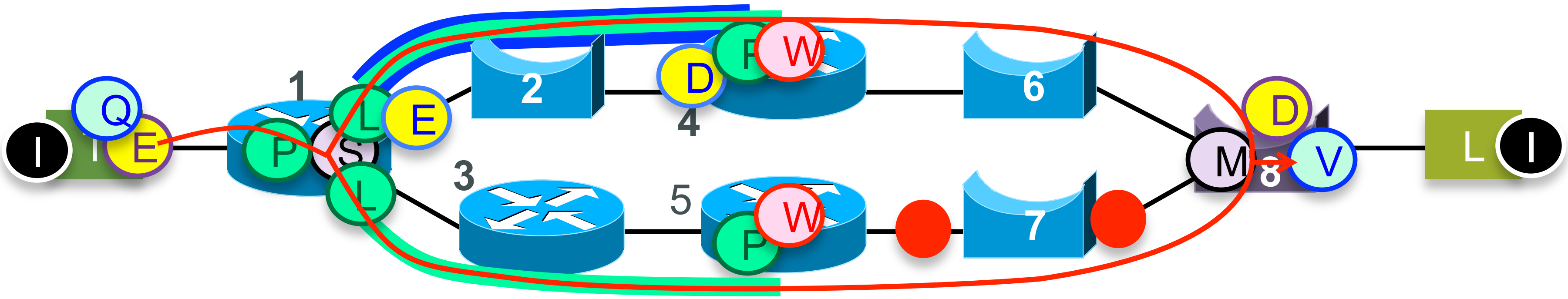
A day in the life of a packet



DA: TSN 7840
SA: Router 4
VLAN tag 23
ET: TSN Seq
Sequence #
ET: IP
IPgram

- So, IPgram pseudowire label 419 is translated by the Interworking function W into TSN circuit 7840[23].

A day in the life of a packet



DA: TSN 12

SA: Router 5

VLAN tag 50

ET: TSN Seq

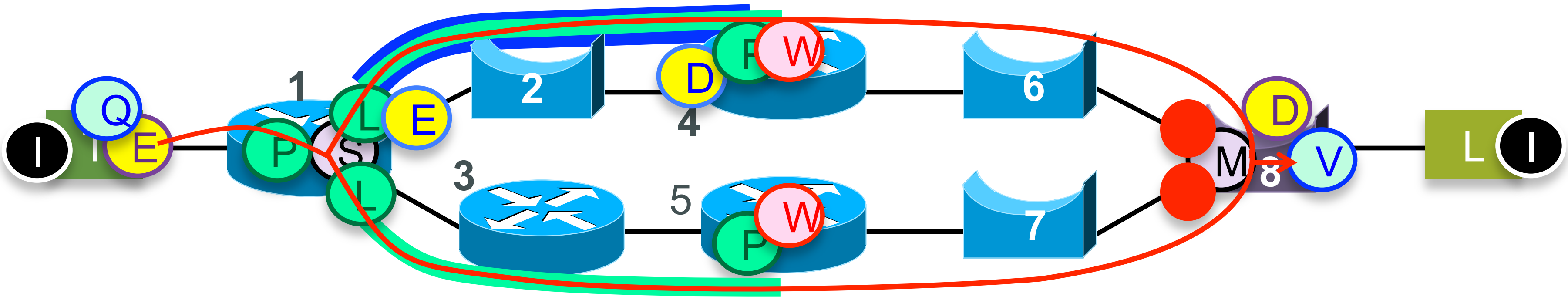
Sequence #

ET: IP

IPgram

- And IPgram pseudowire label 346 is translated by Router 5’s Interworking function W into TSN circuit 12[50].

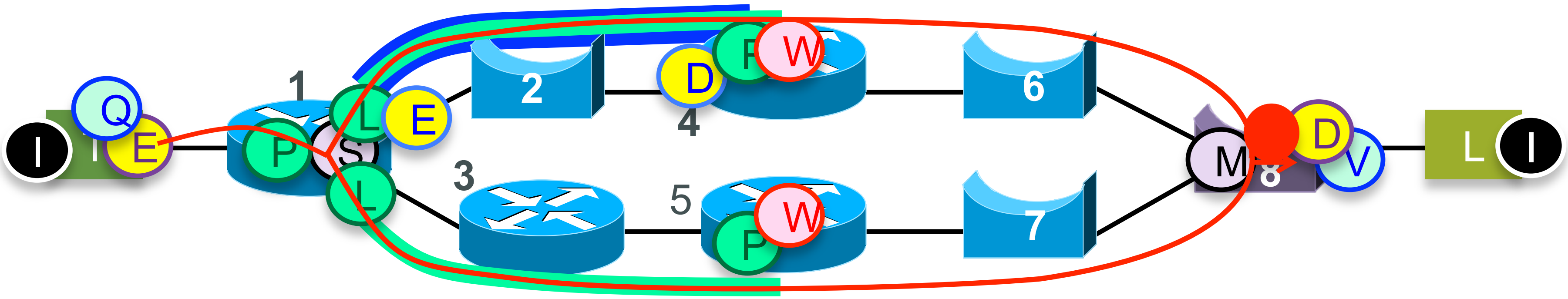
A day in the life of a packet



DA: TSN 7840 or 12
SA: Router 4 or 5
VLAN tag 23 or 50
ET: TSN Seq
Sequence #
ET: IP
IPgram

- The Merge function Ⓜ has to operate on the circuit ID (MAC DA) and sequence number (in TSN tag).

A day in the life of a packet



DA: TSN 734

SA: Router 4

vlan_identifier 99

ET: TSN Seq

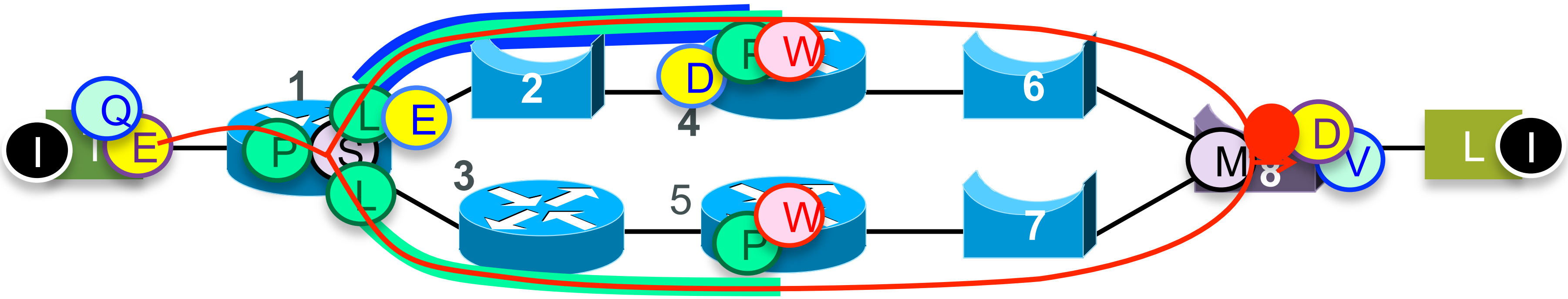
Sequence #

ET: IP

IPgram

- Output from Merge function M
- Note that TSN 7840[23] and TSN 12[50] were combined into TSN 734[99].
- **To Bridge 8**, this is the end-to-end circuit from Talker T.

A day in the life of a packet



DA: TSN 734

SA: Router 4

vlan_identifier 99

ET: TSN Seq

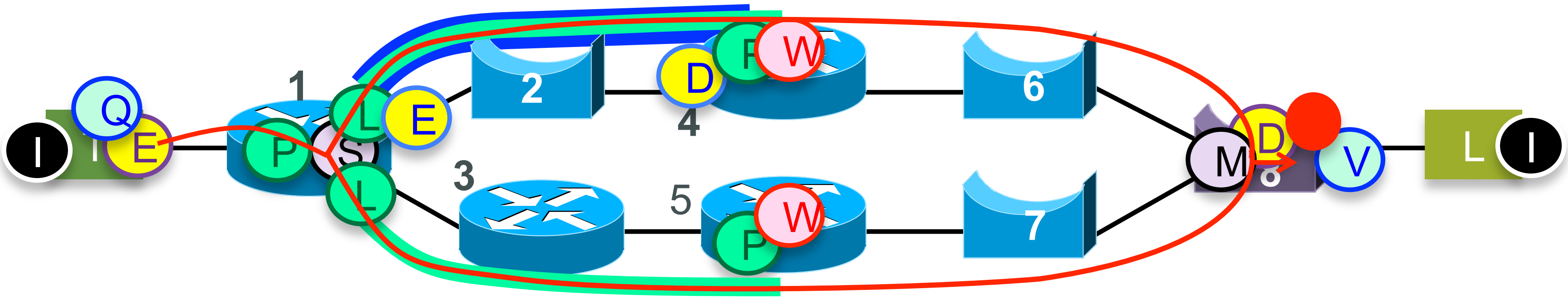
Sequence #

ET: IP

IPgram

- Note that, in this example, the Merge function M passed the packet from Router 4, not the one from Router 5.

A day in the life of a packet



DA: L

SA: Router 4

vlan_identifier 80

ET: TSN Seq

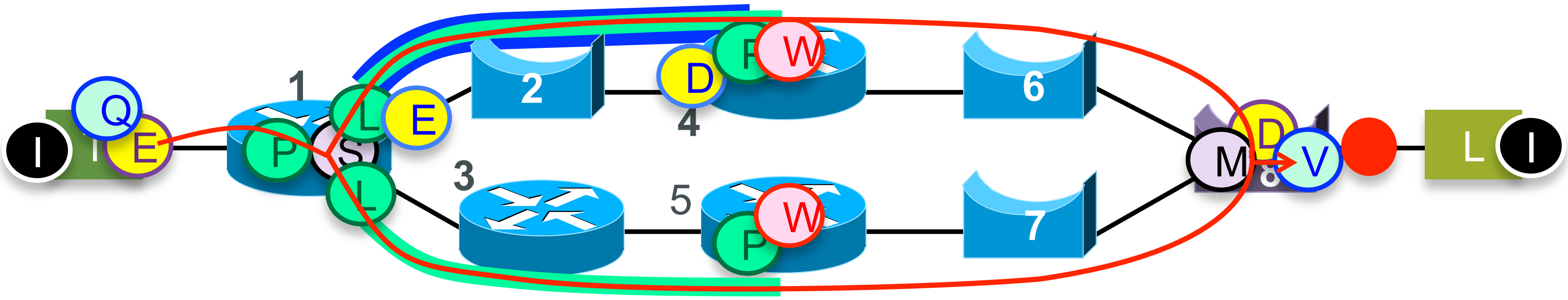
Sequence #

ET: IP

IPgram

- The TSN Decaps function D then replaces the TSN circuit ID with the proper L2 information.

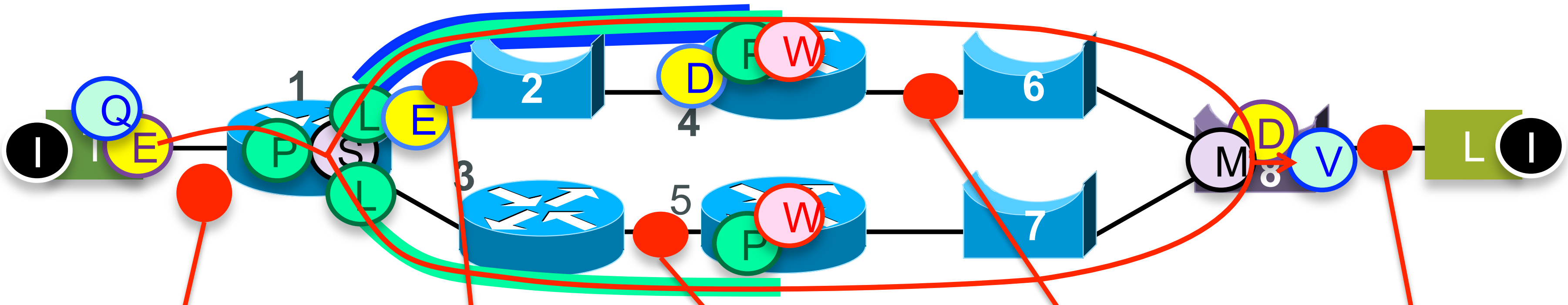
A day in the life of a packet



DA: L
SA: Router 4
VLAN tag 80
ET: IP
IPgram

- Output from Sequenceing function V is what would have been output from an Ethernet Bridged Talker, modulo the VLAN tag changes the bridges would make.

A day in the life of a packet: SUMMARY



- DA: Router 1
- SA: T
- ET: MPLS
- Pseudowire 28
- control (seq)
- IPgram

- DA: TSN 140
- SA: Router 1
- VLAN tag 309
- ET: MPLS
- Tunnel 51
- Pseudowire 449
- control (seq)
- IPgram

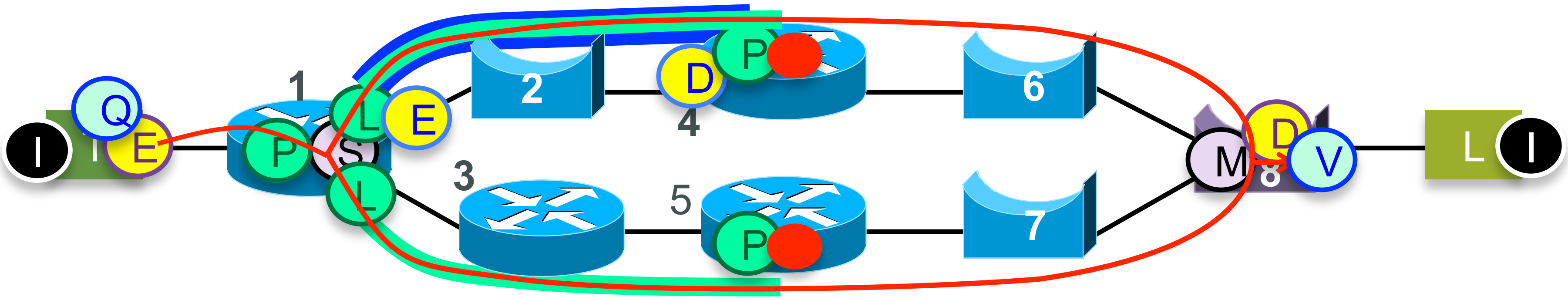
- DA: Router 5
- SA: Router 3
- ET: MPLS
- Tunnel 346
- Pseudowire 31
- control (seq)
- IPgram

- DA: TSN 7840
- SA: Router 4
- VLAN tag 23
- ET: TSN Seq
- Sequence #
- ET: IP
- IPgram

- DA: Listener L
- SA: Router 4
- VLAN tag 80
- ET: IP
- IPgram

Alternatives

Alternative 3: end-to-end pseudowire



●

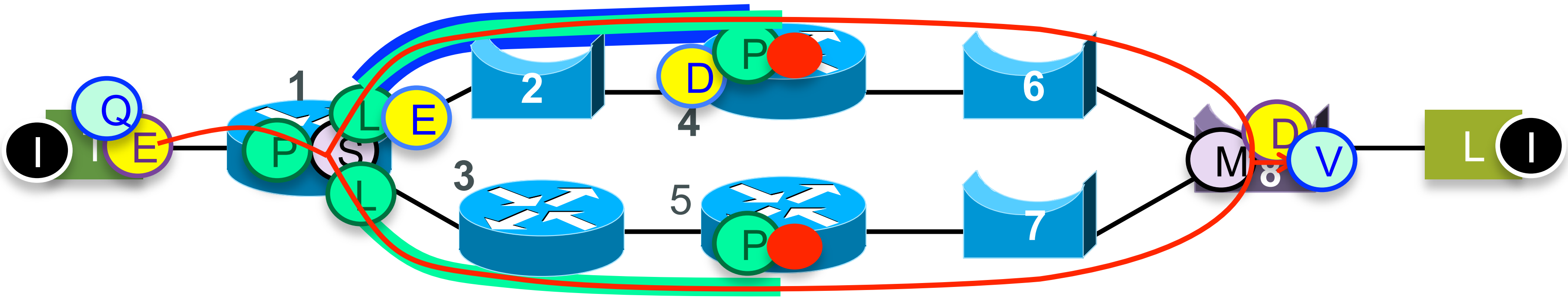
pseudowire label 419
control (sequence)
IPgram

●

pseudowire label 31
control (sequence)
IPgram

- At this point in the preceding discussion, we have the “naked” pseudowire in Routers 4 and 5.

Alternative 3: end-to-end pseudowire



●

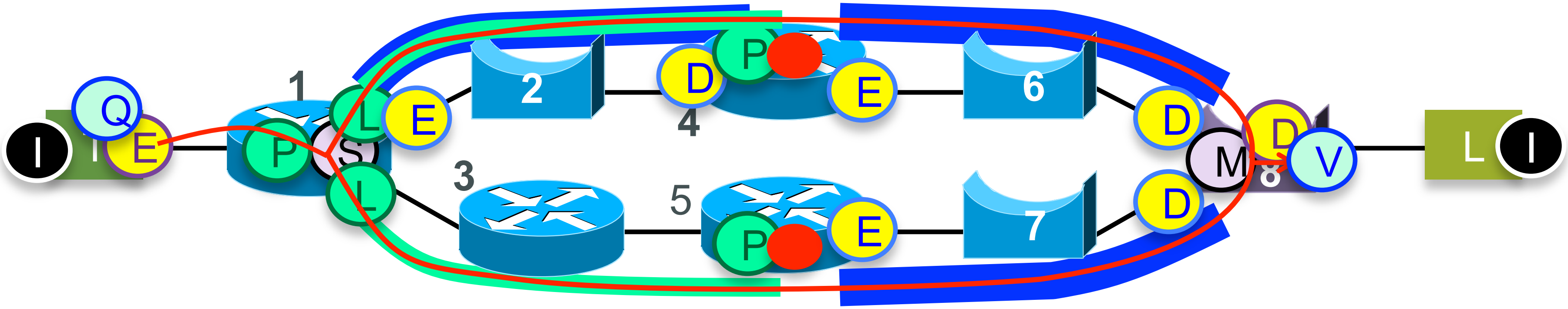
pseudowire label 419
control (sequence)
IPgram

●

pseudowire label 31
control (sequence)
IPgram

- Instead of using an interworking function, we can carry the pseudowire along using the normal MPLS Ethernet encapsulation.

Alternative 3: end-to-end pseudowire



●

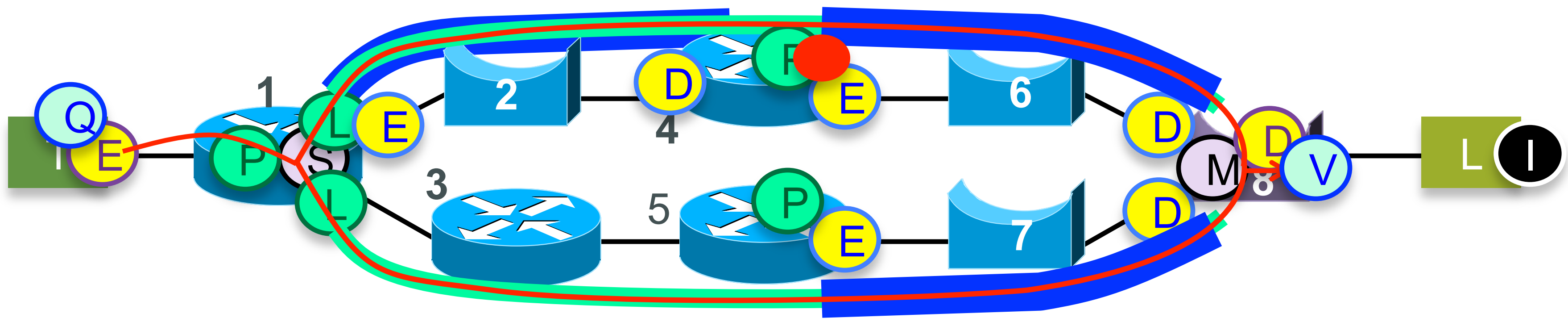
pseudowire label 419
control (sequence)
IPgram

●

pseudowire label 31
control (sequence)
IPgram

- To do this, we need an extra pair of TSN Circuit Encaps/Decaps functions, D E but without an extra Sequencing function Q V.

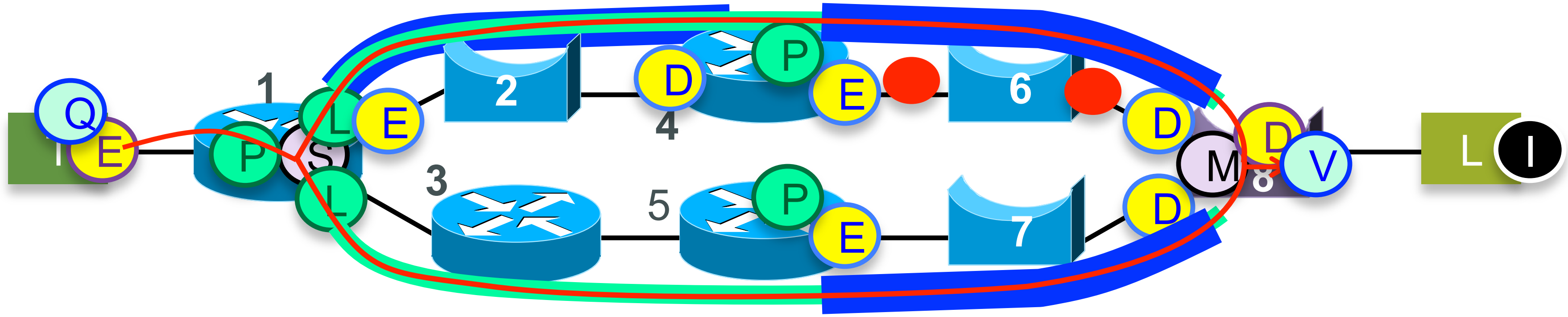
Alternative 3: end-to-end pseudowire



DA: L
SA: Router 4
vlan_identifier 64
ET: MPLS
pseudowire label 419
control (sequence)
IPgram

- Router 4 would have output this frame, if the right-hand L2 network was not a TSN network.
- (Router 5 would be sending something very similar.)

Alternative 3: end-to-end pseudowire



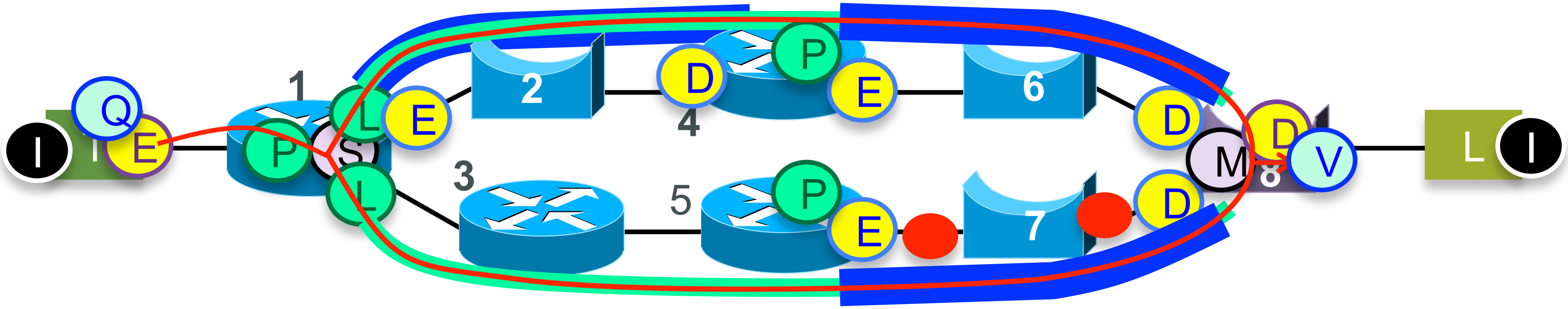
DA: TSN 994
SA: Router 4
VLAN tag 7
ET: MPLS
pseudowire label 419
control (sequence)
IPgram

- But, it is a TSN network, so Router 4 has a TSN Encaps/Decaps function, and generates this, instead.

Note on label stacks

- Apparently, most Cisco hardware, when doing PHP, would determine the MAC DA based on the tunnel label (51 or 346, in this example), not the pseudowire label (419 or 31).
- This should not be a problem. Conceptually, the outer LSP connects Talker T to Bridge 8. PHP simply cuts out the label, for a while.

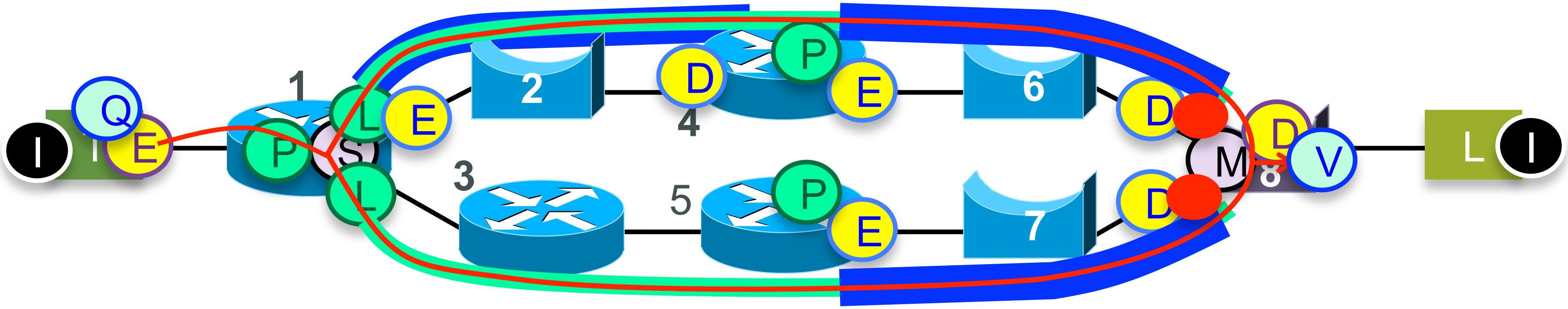
Alternative 3: end-to-end pseudowire



DA: TSN 2006
SA: Router 5
VLAN tag 7
ET: MPLS
pseudowire label 31
control (sequence)
IPgram

- From Router 5 to Bridge 8, the frame looks like this.
- It has a different DA between Router 5 and Bridge 8. The VLAN tag could be different or not.

Alternative 3: end-to-end pseudowire

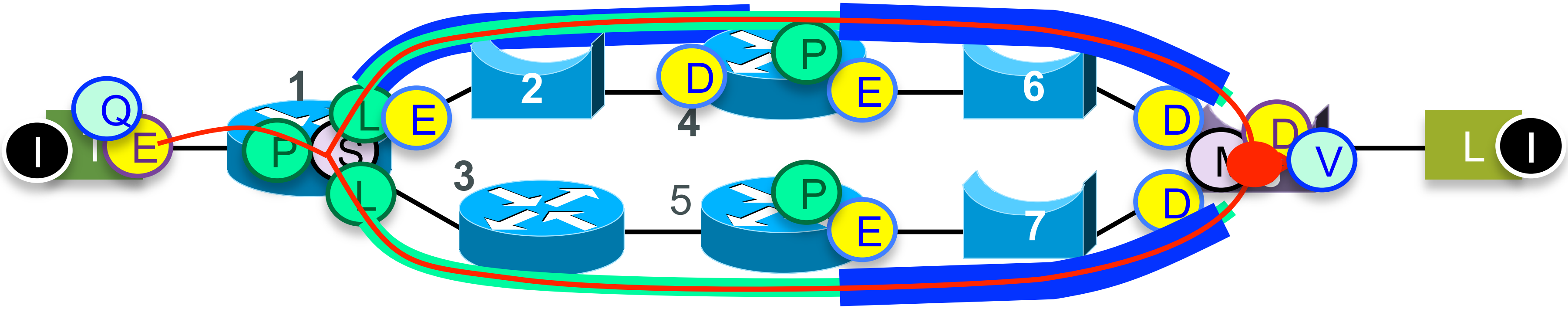


pseudowire label 419
control (sequence)
IPgram

pseudowire label 31
control (sequence)
IPgram

- The Merge function M receives two packets, after the two TSN Decaps functions D in Bridge 8.
- The Merge function also performs the “fools paradise” check, which reports an error if M isn’t seeing mostly the same number of packets on both paths.

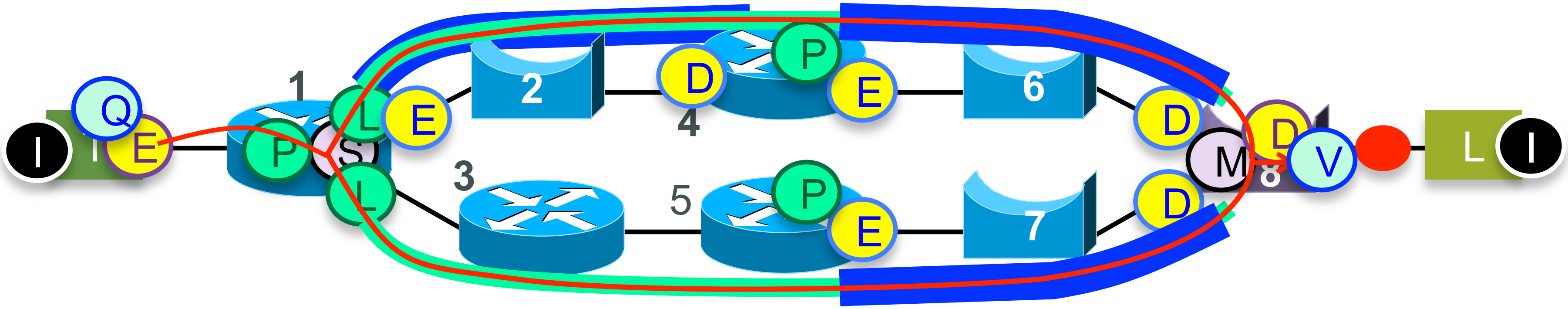
Alternative 3: end-to-end pseudowire



pseudowire label 28
control (sequence)
IPgram

- The Merge function M stitches pseudowires 419 and 31 to produce a single output.
- Note that label 28 is the same label that originated from the Talker.

Alternative 3: end-to-end pseudowire



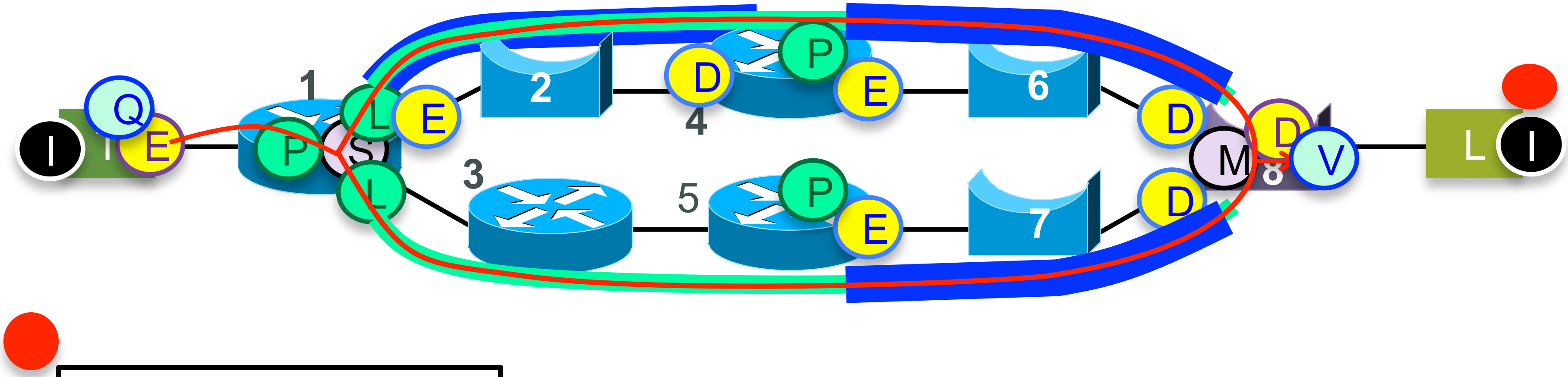
DA: Listener L
SA: Router 4
ET: IP
IPgram

- Bridge 8’s TSN Decaps D and Sequencing V functions terminate the pseudowire, leaving the original IPgram.
- Pseudowire label 28 gets a MAC header with Router 4’s source MAC (not Router 5’s).

Two possibilities

- The Merge function could generate a single circuit (28, in this example) or it could pass both circuits (449 and 31) after deleting the extras.
- If it passed both circuits, then it could supply the “right” router’s source MAC address, depending on which router the packet passed through.
- This is a nit.

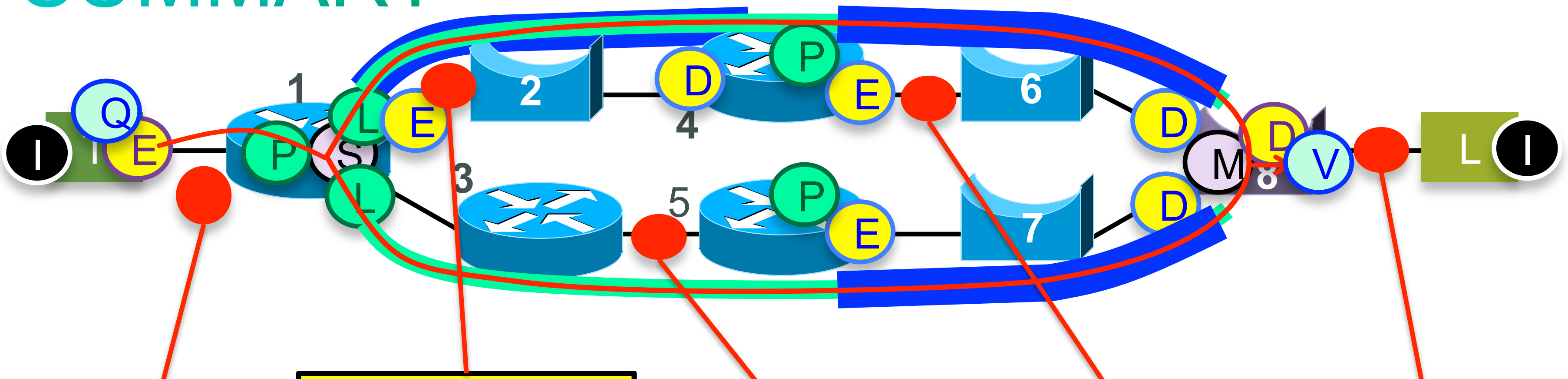
Alternative 4: end-to-end pseudowire



IPgram

- And finally, the original IPgram is delivered up the stack in Listener L.

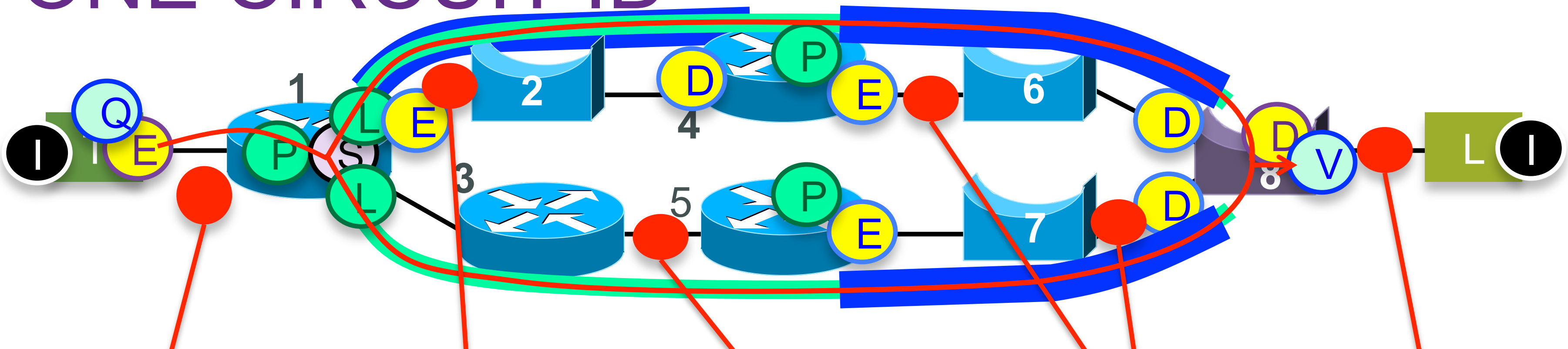
Alternative 4: end-to-end pseudowire SUMMARY



	DA: TSN 140			
	SA: Router 1	DA: Router 5	DA: TSN 2006	
DA: Router 1	VLAN tag 309	SA: Router 3	SA: Router 4	
SA: T	ET: MPLS	ET: MPLS	VLAN tag 7	
ET: MPLS	Tunnel 51	Tunnel 346	ET: MPLS	DA: Listener L
Pseudowire 28	Pseudowire 449	Pseudowire 31	Pseudowire 449	SA: Router 4
control (seq)	control (seq)	control (seq)	control (seq)	ET: IP
IPgram	IPgram	IPgram	IPgram	IPgram

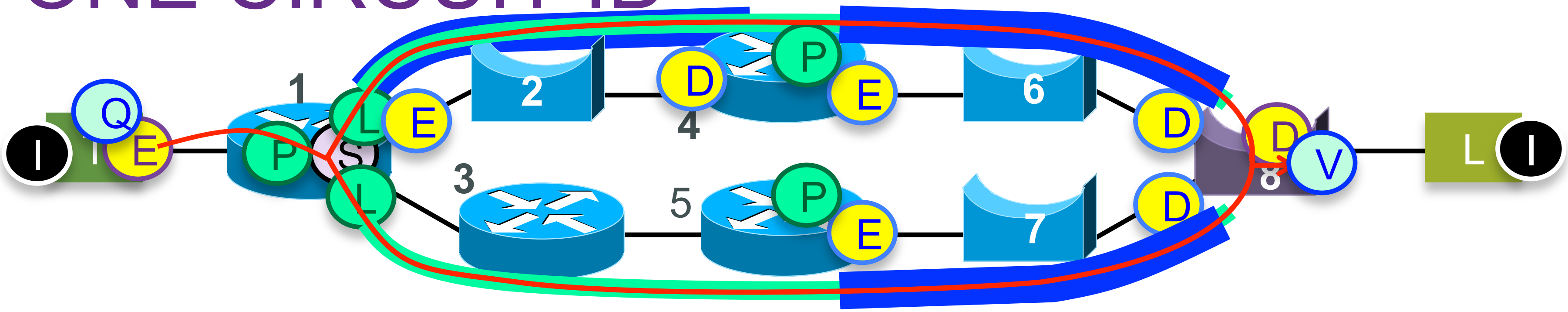
Alternative 5: end-to-end pseudowire

ONE CIRCUIT ID



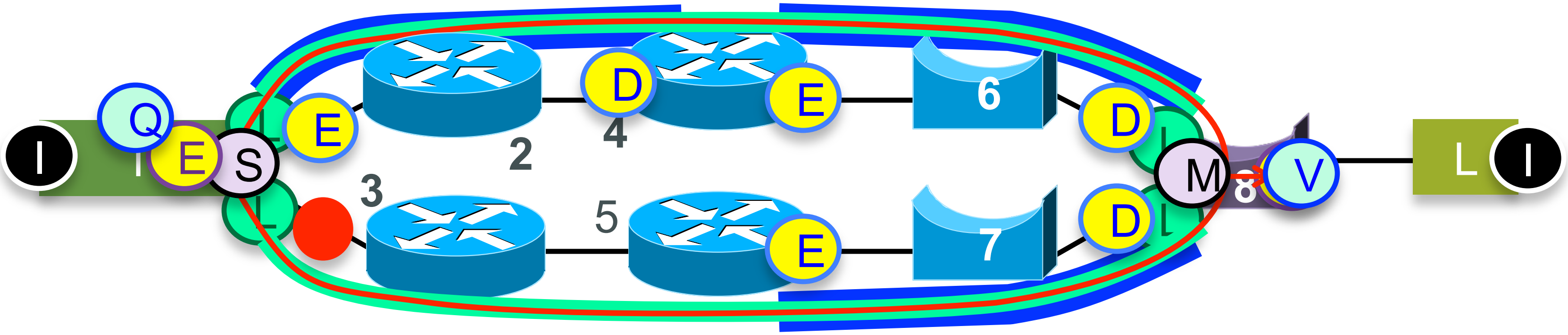
	DA: TSN 140			
	SA: Router 1	DA: Router 5	DA: TSN 2006	
DA: Router 1	VLAN tag 309	SA: Router 3	SA: Router 4	
SA: T	ET: MPLS	ET: MPLS	VLAN tag 7	
ET: MPLS	Tunnel 51	Tunnel 346	ET: MPLS	DA: Listener L
Pseudowire 28	Pseudowire 28	Pseudowire 28	Pseudowire 28	SA: Router 4
control (seq)	control (seq)	control (seq)	control (seq)	ET: IP
IPgram	IPgram	IPgram	IPgram	IPgram

Alternative 5: end-to-end pseudowire
ONE CIRCUIT ID



- Note that the Split function S is still present, in this case, because pseudowire duplication is not a function that is built into the data plane. It does not create new pseudowire labels, though.
- No explicit Merge function M is required.

Alternative 6: Dual-homed Talker



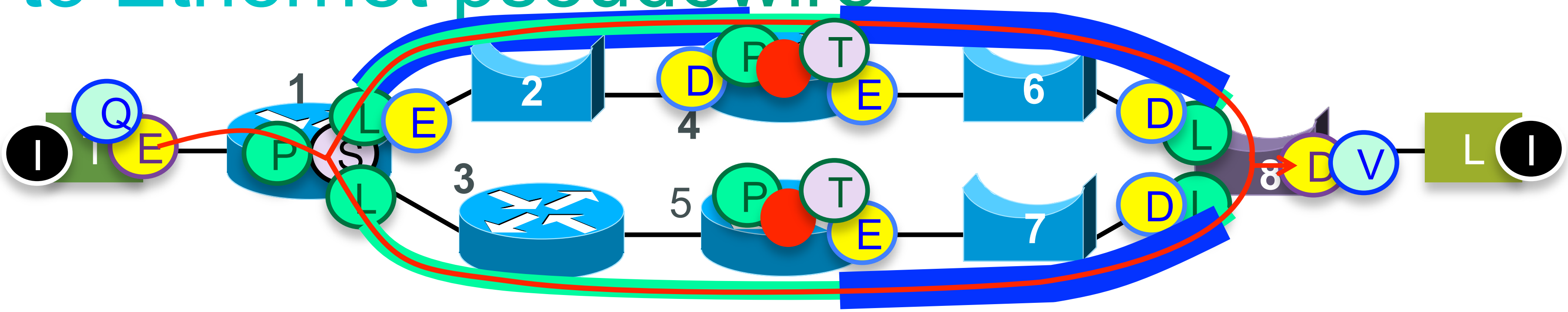
pseudowire label 28

control (sequence)

IPgram

- Talker T could be dual-homed.
- In this case, clearly T must supply the sequence numbers.
- The sequence numbers are usually part of the encapsulation.
- So, T terminates the pseudowire, not routers 2 and 3.

Alternative 7: Stitching IPgram pseudwire to Ethernet pseudowire



●

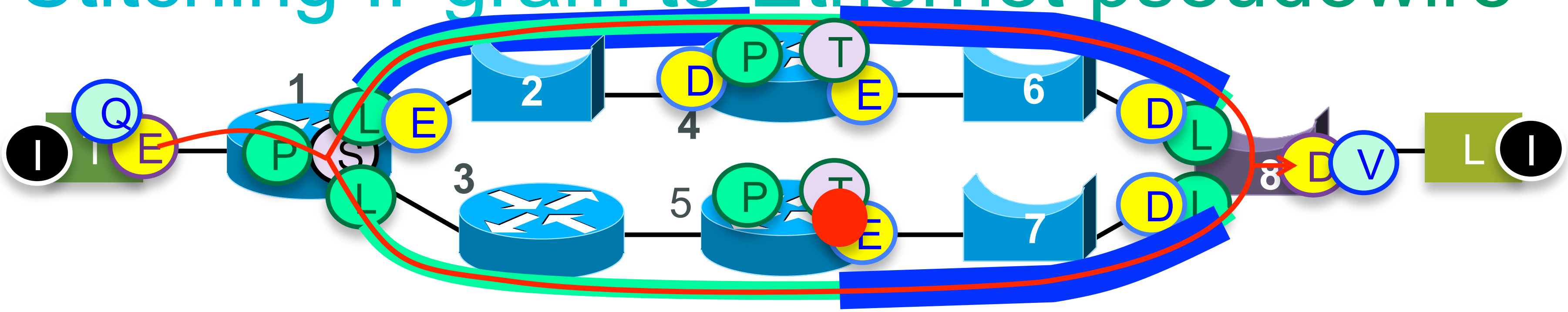
Tunnel label 51
pseudowire label 28
control (sequence)
IPgram

●

Tunnel label 346
pseudowire label 28
control (sequence)
IPgram

- At this point, we introduce a new function: the IPgram / Ethernet pseudowire sticthing function Ⓣ .
- (We're assuming the same label for the pseudowires.)

Alternative 7: Stitching IPgram to Ethernet pseudowire

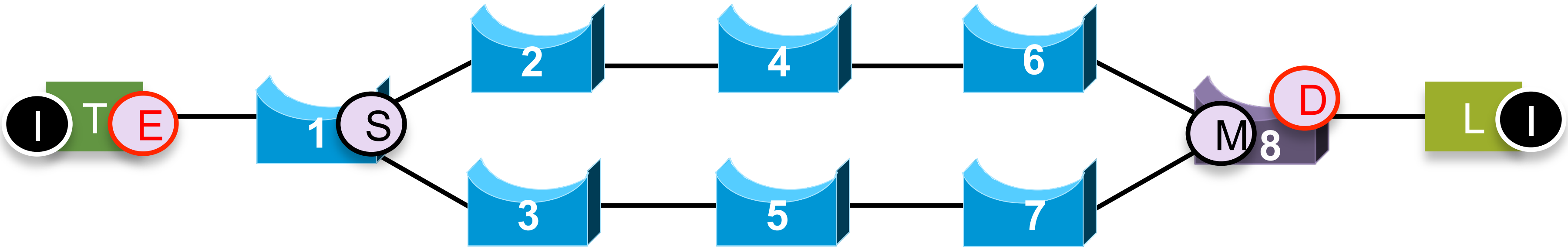


Tunnel label 346
pseudowire label 28
control (sequence)
DA: L
SA: Router 5
ET: IP
IPgram

- The stitching function T converts the IPgram pseudowire to an Ethernet pseudowire in exactly the format to be output to the Listener (or vice-versa).
- In particular, the sequence number is carried through.

Case 3:
Layer 2 end-to-end HSR or PRP tagging

Layer 2 only: **HSR tagging**

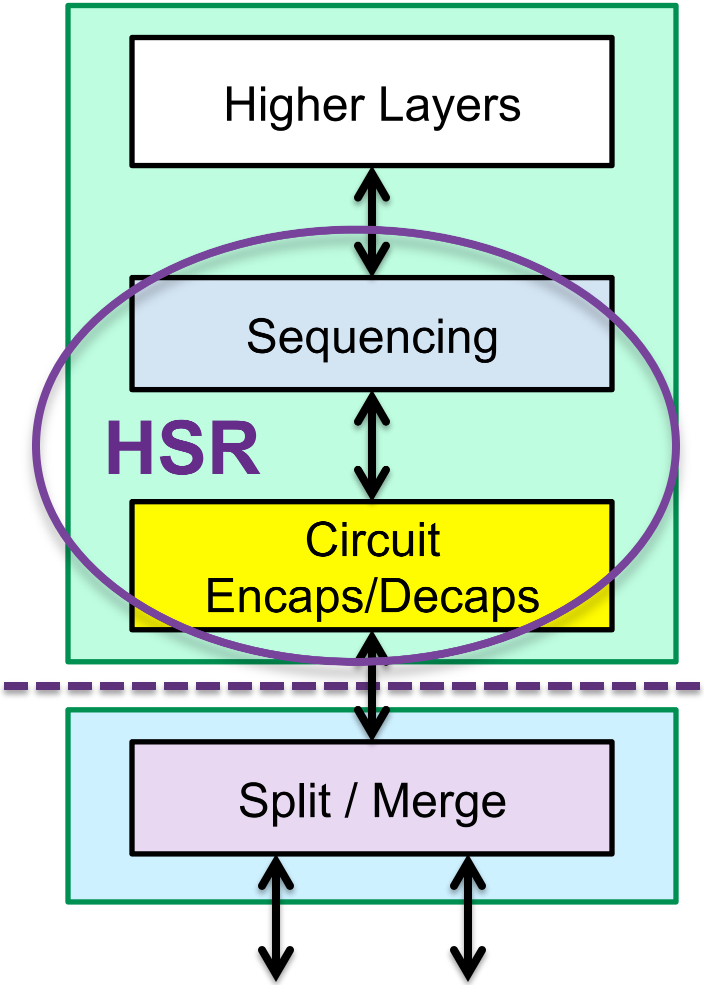


- Again, Talker is TSN-aware, Listener is not.
- This time, Talker is **not** VLAN-aware, Listener **is** VLAN-aware.
- In this case, HSR and TSN Encaps E and Decaps D are combined into a single layer.

HSR-like, not HSR

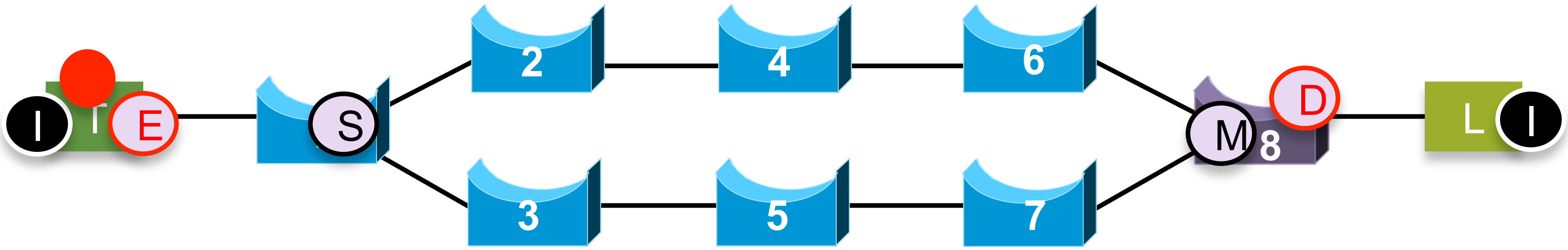
- This is not HSR. It is the HSR format used for a different purpose. This idea may or may not sit well with IEC TC65X.
- This "HSR-like" layer:
  - Connects to a single port, not two.
  - May use one sequence number variable per circuit, not one per host. (This is debatable.)
  - If the station is VLAN aware, has the VLAN tagging below (outside) the HSR sublayer.

A day in the life of a packet



- Note that this is the layering – the top box is Talker T, and the bottom box is Bridge 1.
- HSR combines the Circuit Encaps/Decaps and Sequencing functions.
- It also encapsulates the destination MAC address which, as we will see, is not really very useful.

Layer 2 only: **HSR tagging**



DA: L

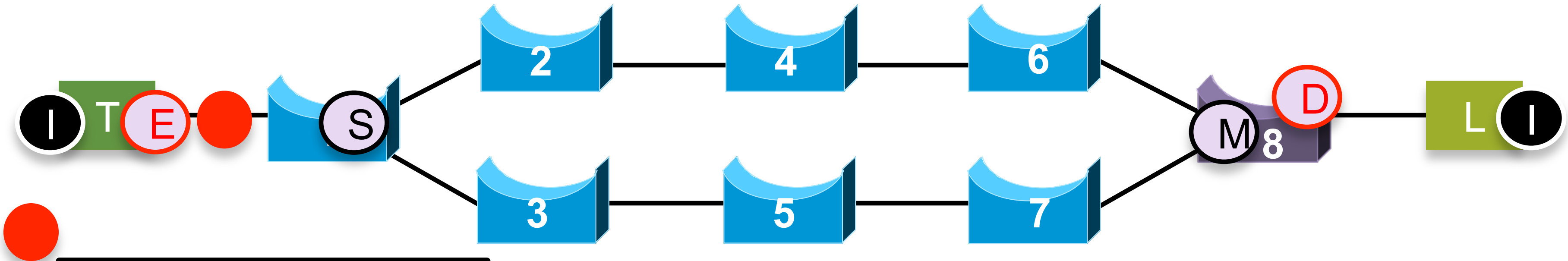
SA: T

circuit_identifier

ET: IP
IPgram

- Talker’s stack is not VLAN-aware. This is what the frame is when it hits the TSN Encaps layer.
- Note that Bridge 1 would normally add a **VLAN 80 tag** to this frame.

Layer 2 only: **HSR tagging**



DA: TSN 734

SA: T

VLAN tag 99

HSR EtherType

pid, length, sequence

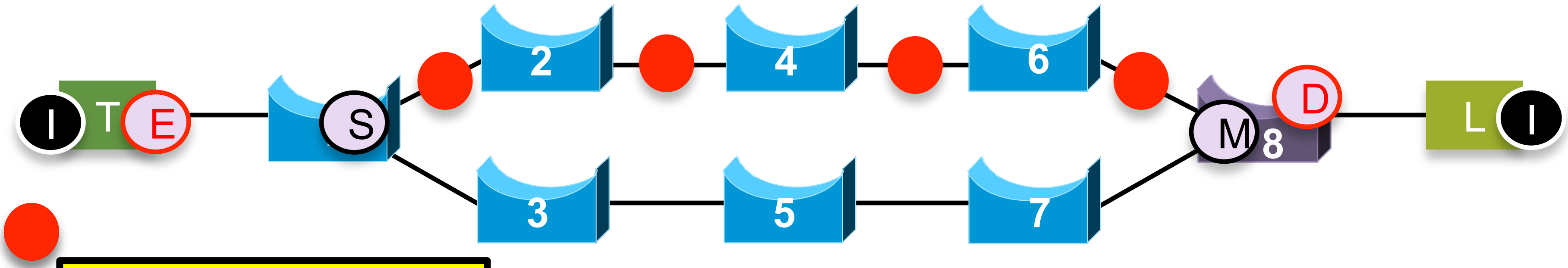
DA: L

EtherType

Data

- The combined HSR/TSN Encaps layer Ⓔ adds a TSN/ HSR tag.

Layer 2 only: **HSR tagging**



DA: TSN 7840

SA: T

VLAN tag 23

HSR EtherType

pid, length, sequence

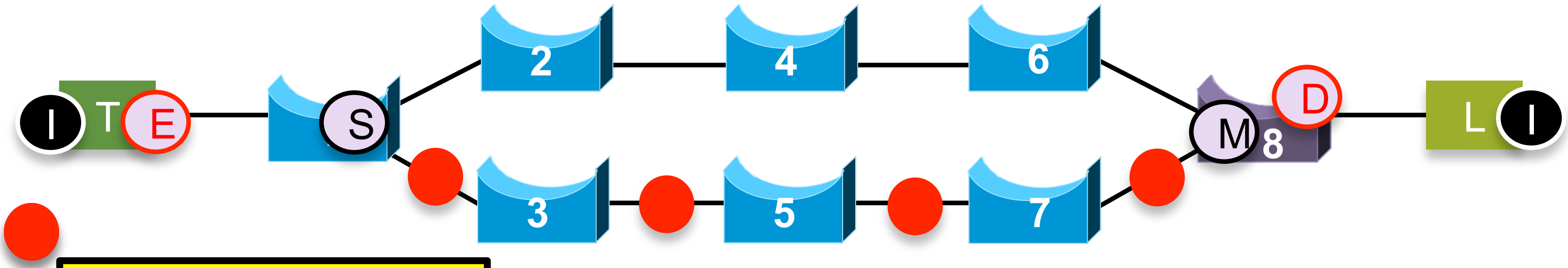
DA: L

EtherType

Data

- The Split function Ⓢ operates on the TSN header, for the path ID, and the HSR header, for the sequence number.
  - (The "pid" field includes a "path A / path B" flag that intended to be different between the two paths. We may or may not follow that usage.)

Layer 2 only: **HSR tagging**



DA: TSN 12

SA: T

VLAN tag 50

HSR EtherType

pid, length, sequence

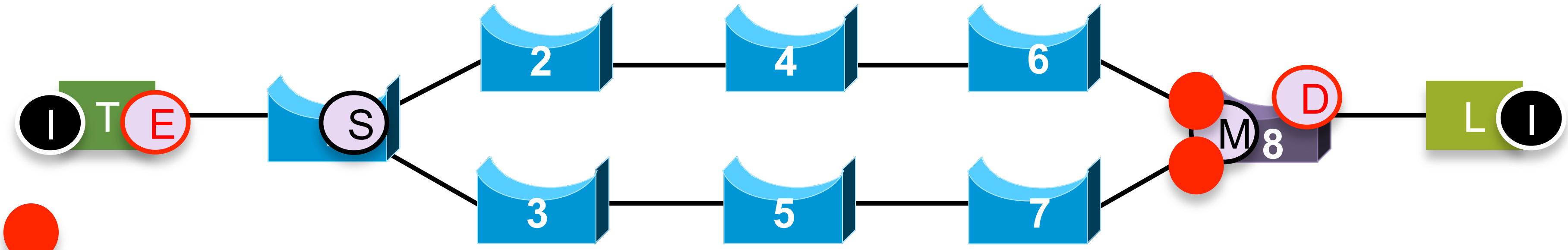
DA: L

EtherType

Data

- The other path gets a different DA and VLAN tag.
- Note that the Split function split TSN 734[99] into TSN 7840[23] and 12[50].

Layer 2 only: **HSR tagging**



DA: TSN 734

SA: T

VLAN tag 99

HSR EtherType

pid, length, sequence

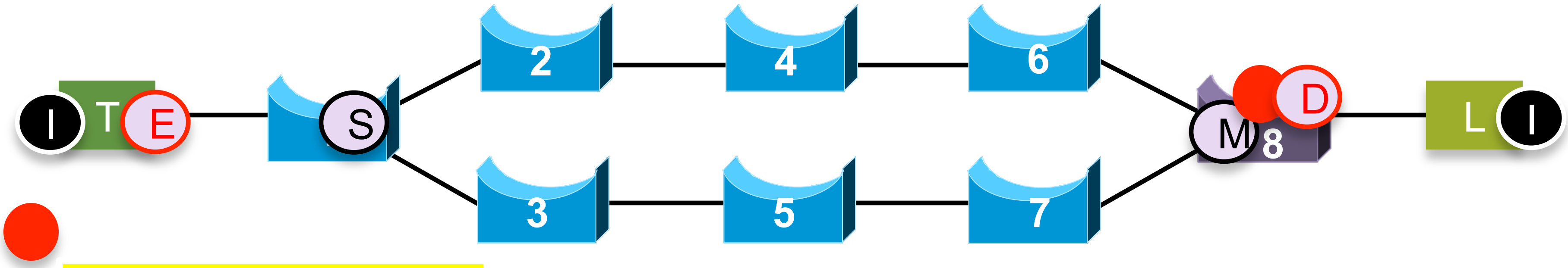
DA: L

EtherType

Data

- The Merge function Ⓜ operates on the TSN header, for the path ID, and the HSR header, for the sequence number.

Layer 2 only: **HSR tagging**



DA: TSN 734

SA: T

vlan_identifier 99

HSR EtherType

pid, length, sequence

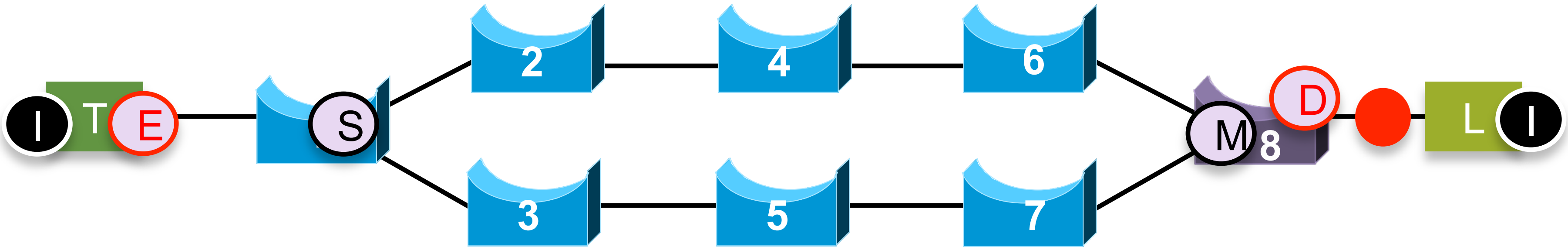
DA: L

EtherType

Data

- Output from Merge function is the original 734[99] tunnel that originated from Bridge 1.

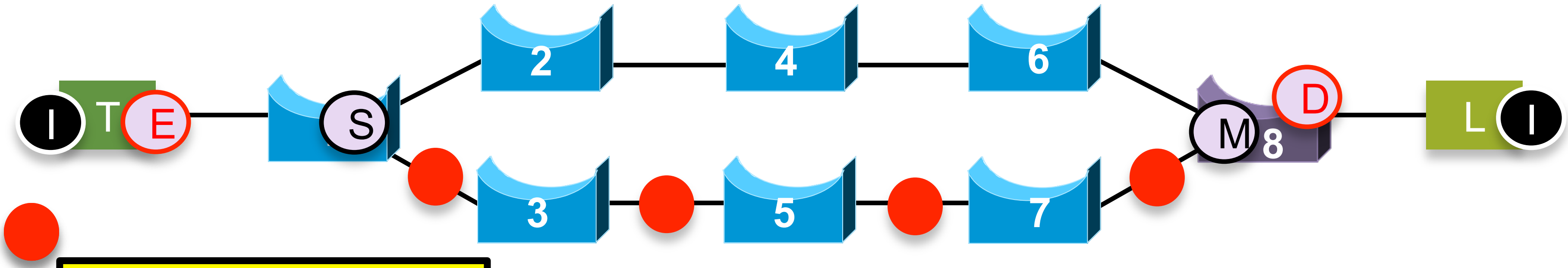
Layer 2 only: **HSR tagging**



DA: L
SA: T
VLAN tag 80
ET: IP
IPgram

- The HSR/TSN Decaps function D, based on knowledge obtained from the control protocol, restores VLAN 80 as a tag.

Layer 2 only: **PRP tagging**

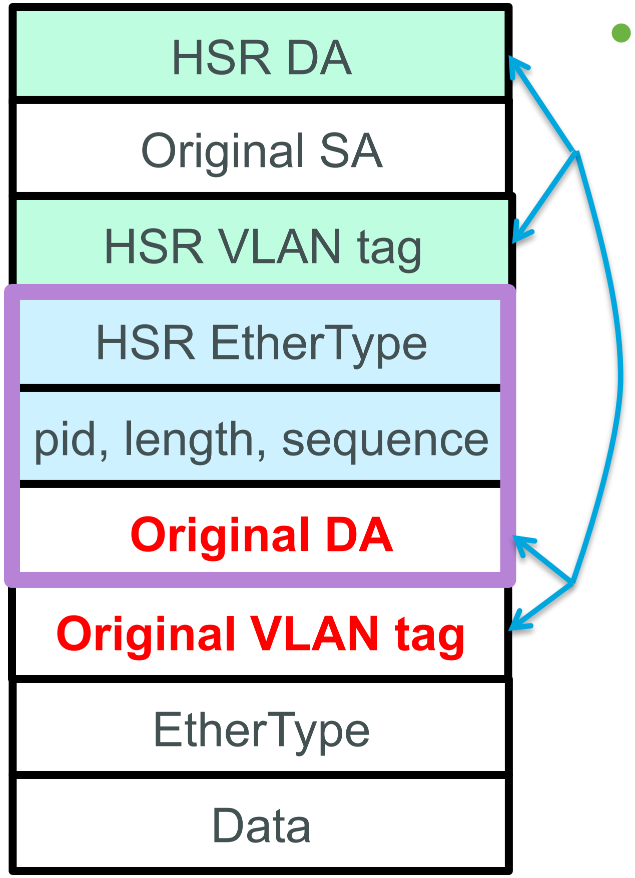


DA: TSN 12
SA: T
VLAN tag 50
DA: L
EtherType
Data
pid, length, sequence
HSR EtherType

- PRP would work similarly.
- This could be useful to interoperate with existing deployments.
- **A big issue with the PRP trailer is that you can’t tell what it’s position is in the tag layering.**

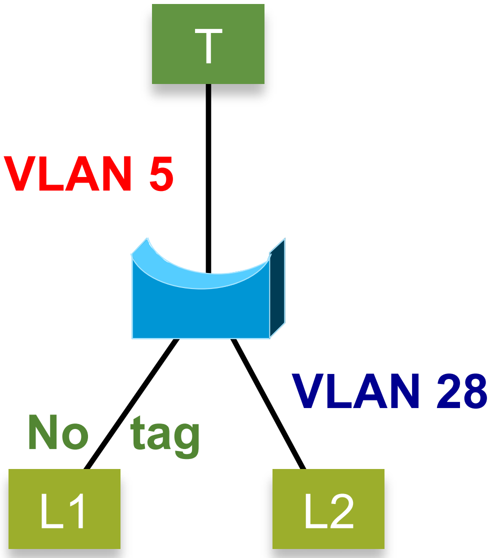
Case 4:
Layer 2 end-to-end Ethernet encapsulation

HSR encapsulates MAC DA and VLAN



- At first glance, the fact that HSR encapsulates the original Destination MAC address and VLAN tag seems attractive, because the decapsulation function can restore the original frame using data in the frame, instead of relying on stored information provided by the control plane.

HSR encapsulates MAC DA and VLAN



- But, in a multicast situation, there may be no one VLAN tag that is suitable for all Listeners.
- Even if VLAN translation in the native Bridge LAN is not present (and it is rare), any Talker or any Listener can be VLAN aware or VLAN unaware.

HSR encapsulates MAC DA and VLAN

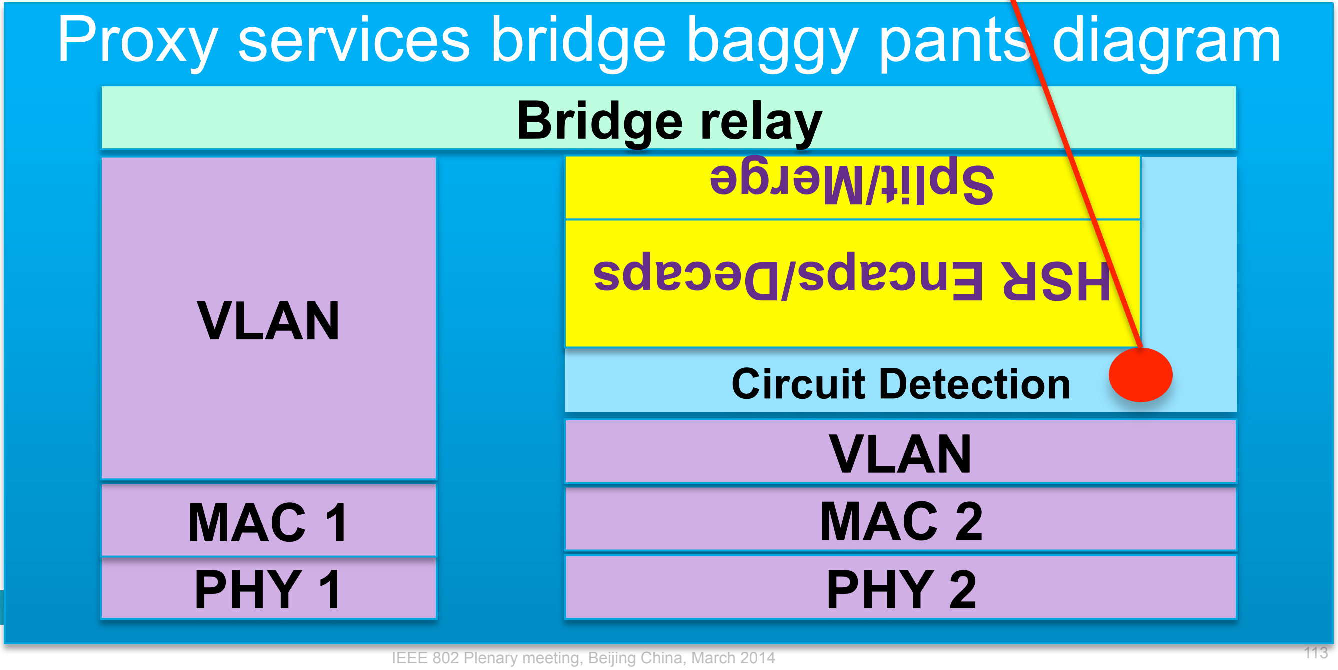
- Also, with HSR, the output up the stack (down, **here**) is a VLAN ID from the Talker's side of the network; it's on the wrong side of the VLAN shim.

Original SA

Original DA

Original VLAN ID

EtherType
Data



Sequenced TSN/pseudowire are simpler

DA: TSN 734

SA: T

vlan_ID 99

ET: TSN Seq

Sequence #

ET: whatever

data

DA: TSN 2006

SA: Router 4

vlan_ID 7

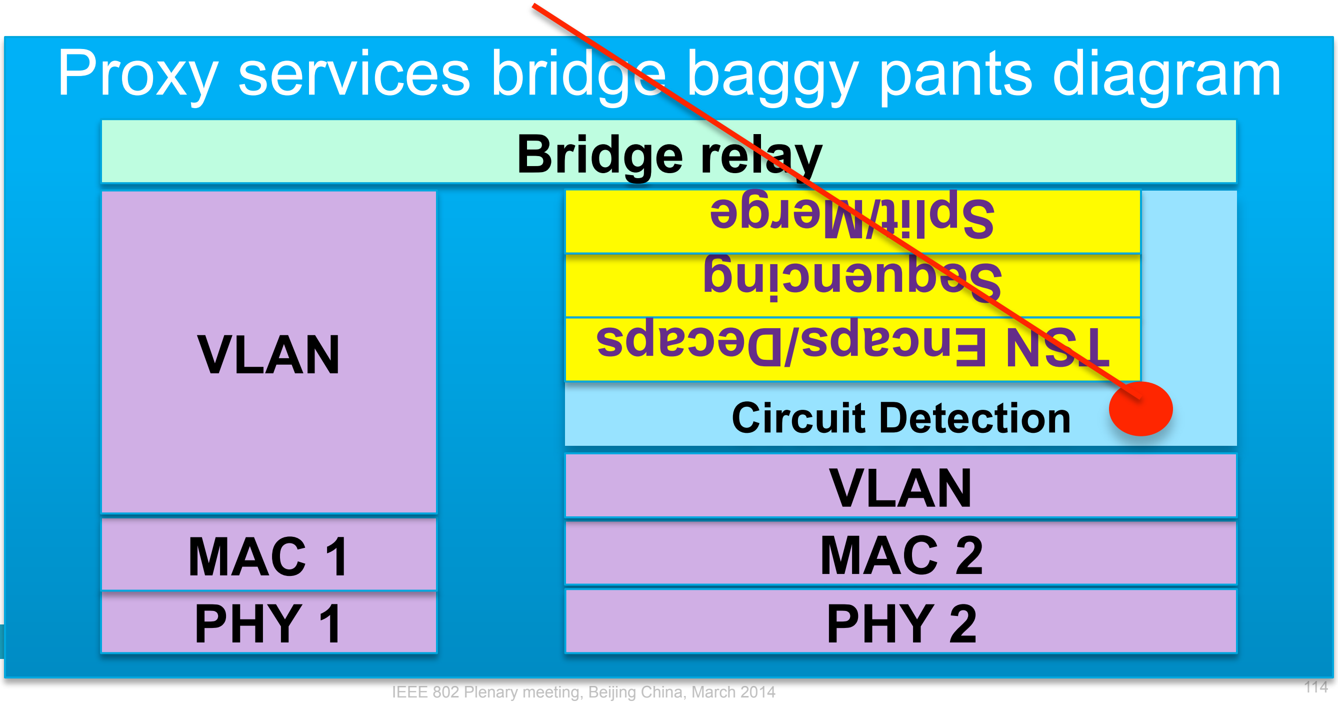
ET: MPLS

Pseudowire 449

control (seq)

IPgram

- Sequenced TSN and IPgram pseudowires are simpler, because they regenerate, not encapsulate. (VLAN ID is generated from control.)



HSR encapsulates MAC DA and VLAN

- So, the VLAN encapsulation does not make things any simpler in the general case.
- If the VLAN tag often needs to be added/dropped/altered, you don’t gain much by encapsulating the DA.
- My conclusion is that, while an HSR encapsulation could be used in some circumstances, it is not the right encapsulation for standardization by IEEE 802.1.

Other end-to-end encapsulations (e.g. PBB-TE or Ethernet psdudowires)

- There are three obvious ways to encapsulate Ethernet frames end-to-end:
  - HSR (does not encapsulate source MAC addr).
  - Ethernet pseudowire.
  - PBB-TE.
- Given the preceding discussions, it is left as an exercise to the reader to see how this can be made work. Certainly, it can be.
- **However …**

However …

- They all suffer from the same issue:
  1. The VLAN ID can be changed as it goes through a Bridged LAN.
  2. The bridge ports to different hosts can be tagged or untagged for different VLANs.
  3. The hosts must have a nagive Layer 2 relationship.
- Furthermore, in the mixed L2/L3 case, tunneling an Ethernet frame end-to-end is risky. There may be locally administered MAC addresses that conflict, and the service cannot be made transparent; the Talker and Listener have no L2 relationship to emulate.

However …

- Therefore, it would seem that the function that decapsulates the Ethernet frame must, at least, be able to insert/remove/alter a VLAN tag, based on information received from the control plane.
- Given that, the difference between end-to-end encapsulation of Ethernet and an L3 encapsulation (IPgram pseudowire) or an L2 translation (TSN encaps) is one of degree (2 MAC addresses and a VLAN vs. just a VLAN added or removed), not kind (altering the packet vs. spitting it out, verbatim).

Summary

Summary

- We have shown how proper layering creates useful packet format possibilities for TSN. This model needs to go into IEEE 802.1 standards.
- There are many more possibilities for creating circuits: VxLAN, LISP, and dozens of as-yet proprietary schemes.
- A new IEEE 802.1 sequence number tag can handle Ethernet end-to-end seamless redundancy.
- Mixed L2/L3 seamless redundancy requires either:
  - Selecting a single end-to-end L2+ split/merge format (e.g. pseudowire); or
  - An interworking function between L3 and L2 split/merge technologies.

Thank you.

CISCO