MAC Status Propagation

Mick Seaman

MAC Relays (such as the P802.1aj TPMR) that provide a frame forwarding (sub)layer below and transparent to other bridges and their protocols can significantly degrade service availability. If the relaying sublayer were not present, changes in connectivity would be accompanied by a change in the MAC_Operational status parameter. MAC_Operational provides rapid notification of connectivity failures and prompts the necessary initial protocol behavior to ensure that new connectivity has not caused an instantaneous data loop. If the MAC status is not propagated by the relays, bridge protocols have to rely on periodic transmissions to detect connectivity changes. These take time to cut loops and repair failures caused by changes in the relayed links.

This note describes media independent propagation of MAC status, while allowing connectivity to a relay while one of its links is not operational. It builds on the ideas discussed at the 802.1 May interim, partly captured in P802.1aj D0.5, and discussed further in the July 2006 meeting. It should be regarded as work in progress.

1. Introduction

The MAC_Operational ISS status parameter (802.1D, 802.1Q clauses 6.4.2) provides a media access method independent indication of the availability of the MAC Service at a service access point (port). The rapid reconfiguration capabilities of RSTP and other protocols depends on link failure notification through a MAC_Operational transition to false. Additions to the underlying physical topology are preceded by MAC_Operational becoming true at a Bridge Port, so that new connections do not result in immediate data loops that a spanning tree protocol will take time to resolve.

A MAC Relay that is not an active participant in the spanning tree protocols should propagate the MAC_Operational state from one port to another. Consider such a relay (Figure 1) connecting two Bridge Ports.

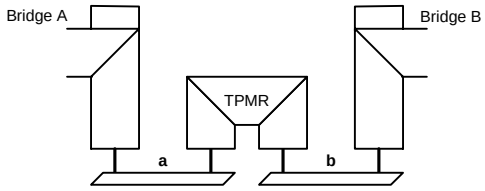


Figure 1—Relay connecting bridge ports

Assume that Bridge A is the spanning tree Designated Bridge for the LAN that comprises the physical LANs **a** and **b** and the TPMR that connects them, and that Bridge B's spanning tree Root Port is shown. If **a** fails (is disconnected etc.) and the TPMR does not relay the MAC status, Bridge B will not reselect its Root Port until it has timed out the last BPDU from A.

Consider a link that uses two relays (Figure 2) perhaps deployed because the intervening 'LAN' **c** is realized using a non-802 technology together with an appropriate convergence function.

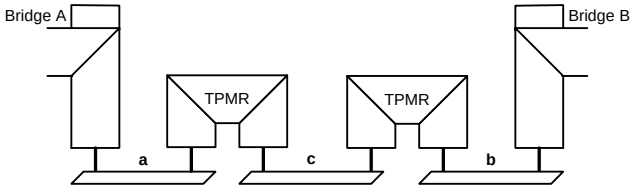


Figure 2—Relay chain connecting bridge ports

Unless the MAC status is propagated, failure of **c** will not be immediately visible to either bridge. Worse, if **c** fails and is restored after a while, both A and B will come to believe themselves to be Designated Bridge for the composite LAN, and will forward frames until one receives a BPDU from the other, even if a data loop has been created†1 †2.

This note:

a) discusses the meaning of MAC_Operational (2)

b) distinguishes 'MAC status propagation', 'link status notification', and 'MAC status notification' (3)

c) describes how management connectivity to relays that use MAC status notification is retained (4)

d) sets out some initial thoughts on the protocol or protocols to support status propagation (5)

e) describes how status propagation should be modelled within the TPMR architecture (6) †3

f) specifies state machines for protocol operation (7)

g) shows that the additional status propagation protocol does not exhibit undesirable properties in cases where it would not have been required, and in cases of mutliple near simultaneous MAC status transitions (8)

h) discusses status propagation for virtual links, such as point-to-point S-VLANs (9).

2. MAC_Operational

To a first approximation, the MAC Service provides (when working correctly) bi-directional connectivity or no connectivity at all. However the fact that MAC_Operational is true does not guarantee connectivity to a peer that is (thought to be) connected to the same LAN. Even if the LAN is point-to-point, the peer could have just reinitialized. Moreover it is clearly not possible, given only the use of the LAN medium for communication, to arrange that two peers

†1It is possible to configure bridge to bridge links so that frames are not forwarded unless BPDUs are received from the peer bridge. A bridge port might be configured to transmit BPDUs always, irrespective of Port Role, or the Designated Bridge could set the Proposal flag in all BPDUs to solicit Agreements continuously (with some impact on network reconfiguration performance). If we do not mandate comprehensive MAC status propagation through TPMRs we should definitely standardize such a solution, even though it depends on configuration, and we may want to standardize it in any case. Of course, BPDU reception monitoring does not speed link failure detection, and so complements rather than replaces MAC status propagation and is not discussed further in this note.

†2This problem (unannounced new connectivity) does not arise if P802.1AE MAC Security is being used and the use of the Uncontrolled (insecure) Port is limited to the necessary authentication and key agreement protocols.

†3TPMRs and the relays described in this note are really just bridges, albeit with possible conformance variations from the bridges standardized so far.

will transition MAC_Operational from false to true (or even from true to false) at *exactly* the same time.

What *is* true for two peers connected to the same LAN is that if MAC_Operational is true for both they can communicate, and if false for either they do not. Protocols, such as the spanning tree protocols, that make use of MAC_Operational to detect new connectivity and initialize state machines rely on the peer that sees MAC_Operational transition last to enforce any necessary special behavior after a status transition†1. For example, it is the last powered on of two connected Bridge Ports that enforces the necessary delay prior to declaring the Port 'OperEdge'.

One way two communicating peers can correctly†2 implement MAC_Operational is to signal to each other when they are ready to receive, and to assert MAC_Operational true and accept frames (for transmission) and deliver frames (on reception) from and to their local client on receipt of 'ready to receive' from their peer†3. This does not require a MAC client to recognize separate 'receive operational' and 'transmit operational' status parameters. Signalling 'I am ready' is equivalent for links that are not aggregated†4 or not expected to be aggregated without an increased risk of frame loss.

3. Status notification and propagation

Clearly a change in the operational status of a given LAN can be communicated by transmitting a frame (over another LAN!) conveying that information. This note defines the term 'link status notification' to describe the transmission of a frame conveying information about (some LAN's) MAC_Operational parameter. The term 'MAC status notification' is used to describe a (layer) management interaction with an underlying MAC Service to change MAC_Operational in a bridge or relay that is a peer user of that service†5.

These two notification methods differ, as follows.

Link status notification does not interrupt or prevent other communication between adjacent relays, or between a relay and a bridge. On the other hand, its successful use depends on both implementing a new protocol. Since it does not prevent communication it cannot, by itself, prevent loops caused by new connectivity.

MAC status notification is the equivalent of cutting the wire. It is immediate, and effective for all MACs and services that currently support MAC_Operational correctly. It also interrupts all other communication, including the use of in-band management to rectify an underlying problem.

This note uses the term 'MAC status propagation' for the overall process of communicating a MAC_Operational parameter value through one or more relays, possibly involving link status notification between some and MAC status notification between others.

4. Management connectivity

If MAC status is propagated using MAC status notification, management connectivity is lost. Link status notification is generally preferred, unless it is known that one party does not implement it, or responds slowly to the receipt of notification frames.

To allow management of links that would otherwise propagate a 'down' MAC status, the MAC status can be 'blipped', i.e. taken false for a brief period whenever it changes, and allowed to return true. While this slows the recognition of newly available connectivity, that is rarely an issues since several seconds hysteresis should be applied to any MAC_Operational status transition to prevent higher layer protocols 'flapping'†6.

5. Initial protocol thoughts

What we seek is not so much *a* new MAC status propagation protocol, as an analysis of required functionality that can be mapped onto existing protocols, with minimal extensions. In particular the functionality provided by CFM should already be close to what we need, while the various MACs all have more or less sophisticated capabilities are part of basic operation. In analyzing the required functionality, we further hope to identify fundamental communication primitives that can be relayed independently of the details of particular protocols—thus providing a overall solution for MAC status propagation on a link that might comprise individual LANs of different MAC types and chains of TPMR like devices with their own preferred link maintenance protocols.

Throughout this protocol description the term 'LAN' will be used solely to refer to the individual LANs at the lowest layer in the architecture shown, connecting adjacent TPMRs or a TPMR and an adjacent bridge. A 'chain' is a series or part of a series of TPMRs connected by LANs, and a 'link' is the connectivity provided by a chain between non-TPMR devices communicating at a higher (sub) layer that see entire link as a simple transparent LAN.

This note makes an underlying and very important assumption. It is concerned about what happens in plug and play and default cases, where TPMRs provide links that are either not subject to detailed provisioning management, or have not been provisioned yet. Of course moving some equipment around is a good way to provoke this scenario. The use of provisioned CFM between two or more selected ports on the TPMR chain, or between the end of a link, cannot be relied upon, though it would be nice if such CFM use fitted neatly into an overall solution. It is quite possible that those who want to rely totally on the fail-proof nature of their pre-provisioning systems and operational practices will want to disable what ever plug-and-play solution we devise to the problems of unsignalled connectivity changes. However we should have a solution for the person who is surprised when his new TPMR deployment starts trashing his network. Because some administrators will turn off whatever protocol we devise to keep them out of trouble, the protocol itself should be benign if there are TPMRs in

†1 Such protocols are, therefore, naturally symmetric in operation.

†2 That is, in a way that exhibits no frame loss after MAC_Operational becomes true for both, unless it very shortly becomes false for one or the other again.

†3 This is essentially the rule used by 802.3ad link aggregation (when implemented properly) to ensure that frames are not lost when links are added to an existing aggregate.

†4 It is not equivalent for a link (LAN) that is being added to an aggregation, since such a link should deliver frames to its 'user' (the aggregator) once it has sent 'receiver ready', but should not be used for transmission until it has received 'receiver ready' from its peer. Clearly both ends of the aggregate link cannot wait for the other to send 'receive ready' first.

†5 I am not wedded to these particular terms but there is a need for three simple terms, one for communication by 'cutting the wire', one for communication using frames, and one for the overall process of communication through concatenated relays—possibly involving transmission of frames between some relays and wire-cutting at others.

†6 It is an open question as to whether the 'MAC' or the higher layer protocol should apply the hysteresis. The latter clearly provides more flexibility, but has less knowledge of when the underlying service is truly likely to become reliable for a sustained period.

the chain that do not use it—though of course it will not then meet its goals for link operation.

The number of TPMRs in the time sequence examples that follow is deliberately high, four where one or two might be more common, in order to show all the necessary aspects of protocol behavior. In the examples, the MAC_Operational and MAC_Enabled status at each port is shown, (thumbs up) for OperUp, (thumbs down) for OperDown but locally MAC_Enabled, (pointing down) for MAC_Enabled false, and (pointing up) for MAC_Operational but temporary preventing traffic through the port by reporting OperDown to the port's clients.

There are two major approaches to the design of protocols of the type investigated in this note. One communicates the status of a link continually, 'up' or 'down', the other is event driven and focuses on the transitions—the addition of connectivity, 'add', or loss of connectivity, 'loss'. It is fairly easy to convert signalling of one type into the other, and both approaches can be made to communicate the state held by the transmitting entity and to be (to within the time deltas implied by message loss or repetition) idempotent. At the same time it is necessary to be very clear what form of signalling is being used, as the design process goes around in circles if metaphors are mixed. The initial presentation in this note uses the event (transition) signalling approach.

This protocol description begins with signalling new or recovered connectivity. This is more difficult than signalling failure—the wide range of options for the latter provide little guidance for the overall design. Figure 3 shows the ideal response to a new LAN connection in a link between two bridges that do not participate in link status protocol—and thus require MAC status notification.

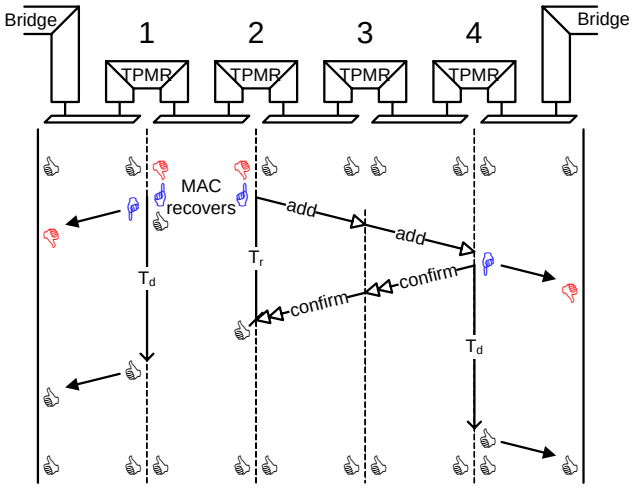


Figure 3—New connectivity

When the new connection is made, the TPMR ports at either end of that LAN recognize OperUp but do not immediately report that to their local clients ( ), thus preventing an immediate data loop. The TPMR (1) that is immediately adjacent to a higher layer bridge disables its other MAC ( ) for a period to signal new connectivity to the bridge. The other TPMR (2) sends an 'add' message on its other port. The 'add' propagates through the chain of TPMRs until it reaches the last LAN, connected to a higher layer bridge. The TPMR's MAC for this LAN is disabled and a 'confirm' message is sent through the chain to the TPMR port that originated the 'add', causing that port to report OperUp to its clients ( ). Finally TPMR 4 re-enables its disabled MAC restoring connectivity to the adjacent higher layer bridge. The time (T_d in the figure) for which that MAC is disabled should be chosen to be sufficient for the 'confirm' to reach the TPMR originating the 'add', and for that TPMR to enable its clients, thus ensuring that there is connectivity between the two higher layer bridges when they both report OperUp to their local protocol clients†1.

The foregoing assumes that TPMR 4 knows that its right-hand LAN connects to a system that does not participate in the link status propagation protocol. If that system did participate, the time for which connectivity from it to the TPMR chain is prevented could be kept brief. However we have to recognize that most existing equipment will never be changed to add status propagation support, and that the benefit is minor as compared to potential competing demands on engineering and deployment time. It is therefore quite likely that it does not participate, and that the TPMR (4) is not aware of that before hand. Figure 4 shows how the protocol deals with this scenario.

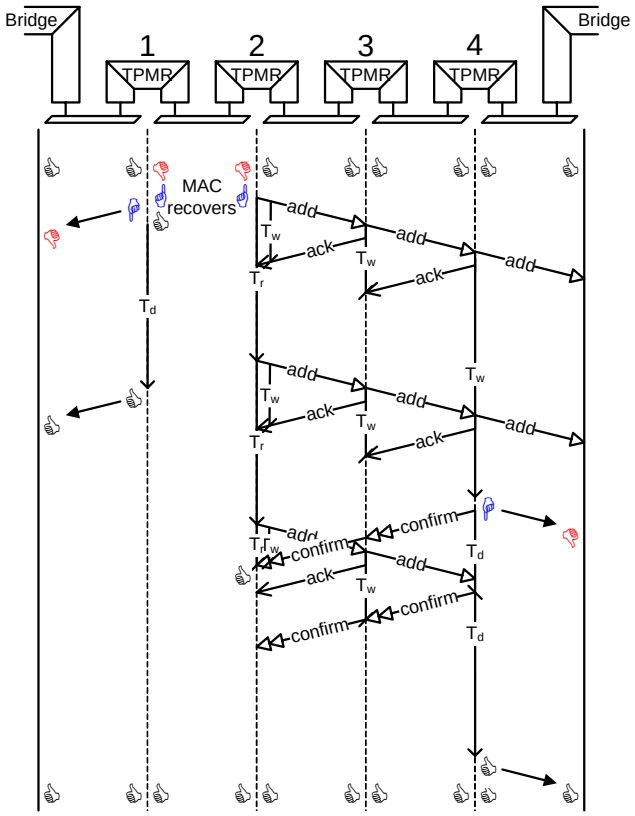


Figure 4—New connectivity with MAC status notification

After receiving, and forwarding, 'add' messages without receiving a confirm, TPMR 4 disables its right hand MAC to be sure that the attached system sees an OperUp/Down/Up transition. Since MAC status notification should only be used at the end of the TPMR chain to avoid disrupting TPMR to TPMR protocols†2, an acknowledgment that is local to two adjacent TPMRs, is used to suppress that transition (by cancelling the T_w timer).

†1 While relying on any time in this way is tacky protocol design, there is no reasonable alternative without requiring the bridge at the end of the link to participate in the protocol. Since link up downs should have ample hysteresis (at least 2 seconds) to protect against flapping links, while the confirm should be forwarded and processed within a few milliseconds, the right result should be easily obtainable without timer tweaking.

†2 Like the spanning tree at the relay level, which will be required inevitably once we allow 'three port MAC relays', communication switching systems that 'look like MAC relays' and so on. Unless of course we want to put the technology back thirty years and invent loop detection protocols.

This ‘ack’ can be a true ack, prompted by receipt of the ‘up’, or any other indication that the next (or a further) TPMR in the chain has or will receive the ‘add’ to within a reasonably low probability of loss—such as having received any message from that TPMR indicating that it is operating the protocol. Similarly there is a degree of flexibility in the choice of when the MAC is to be deliberately disabled to propagate MAC status. That could happen, as shown in Figure 5, after a timeout (Tw) after the receipt of an ‘add’ without a corresponding ‘ack’ or ‘confirm’, or it could happen, as in Figure 3, on receipt of first ‘add’ if the knowledge that the TPMR is at the end of the link status notification chain is already available. The state machines for each TPMR port status propagation can vary in the way that this detail is handled, without affecting the viability of the link status propagation and confirmation along the TPMR chain.

There is little reason not to use an ‘ack’ when signaling connectivity failure as well, see Figure 5. The ‘confirm’ message could have been avoided by passing on the responsibility for retries using the ‘loss’/‘ack’ exchange, as the port that experienced the transition on TPMR (2) is not holding state. However using different propagation philosophies for the ‘loss’ and ‘add’ cases seems odd†1. As it is, it is clearly the case that ‘add’ and ‘loss’ could be replaced by a single ‘chg’ (change) message, which gives more flexibility in adapting to existing protocols.

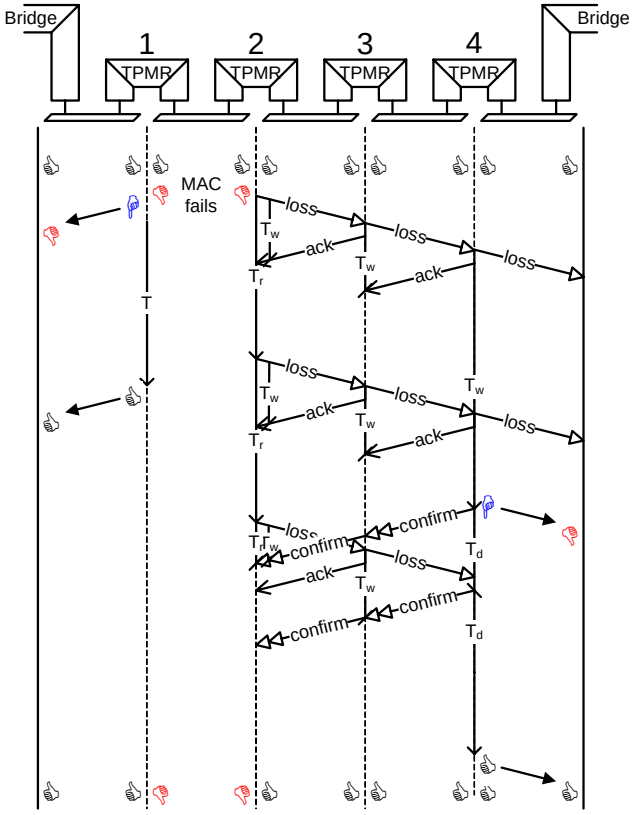


Figure 5—Connectivity failure

†1I originally wrote this up the other way round, but the difference continued to jar.

6. Interface architecture

It is clearly desirable to add MAC status propagation to the existing bridging architecture with the minimum of changes to the existing components of the specification. That is why the protocol summary suggested continuing to report MAC_Operational false to the local client of a port that has just come up, pending status propagation. A shim per port, or at least modelling using a shim, is clearly indicated. Figure 6 shows shims for the two ports of a TPMR, together with a MAC Status Propagation Entity (MSPE) that facilitates communication between the shims †1.

Status propagation through LMI interactions with the MSPE allows the use of generic management requests and indications that can be communicated differently by the different MACs, as well as providing a way for one MAC State Shim to signal to the other(s), through the MSPE, even though MAC_Operational for its upper ISSAP is false. Modeling status propagation through the relay in terms of forwarding link status notification frames would lack that latter attribute.

The MSS does not receive or transmit frames itself, but calls upon the MSPE to do that on its behalf. If the MSS is reporting MAC_Operational False to its clients (ultimately the MAC Relay Entity, the MSPE, and other Higher Layer Entities) it allows neither the reception nor the transmission of frames.

The MSPE does not forward link status notification frames itself, though an LMI request from an MSS can result in frame transmission on another port. Link status notification frames are forwarded by the MAC Relay Entity, like any other frame. This provides the fastest possible communication of status notification. Link status notifications are sent to a multicast address, and are received by the MSPE as well as being forwarded. The MSPE attaches to each Port by a Bridge Port connectivity function (see 802.1Q clause 8.5.1) that is augmented to allow transmission and reception only to the attached individual LAN, as shown in Figure 6.

†1It is probably worth noting that, in a TPMR, the VLAN tagging functions are very much a subset of those currently in 802.1Q Clause 6.7, being limited to tag recognition to support service class queue. Tags are not added, removed, or changed by the TPMR but retaining the tagging function

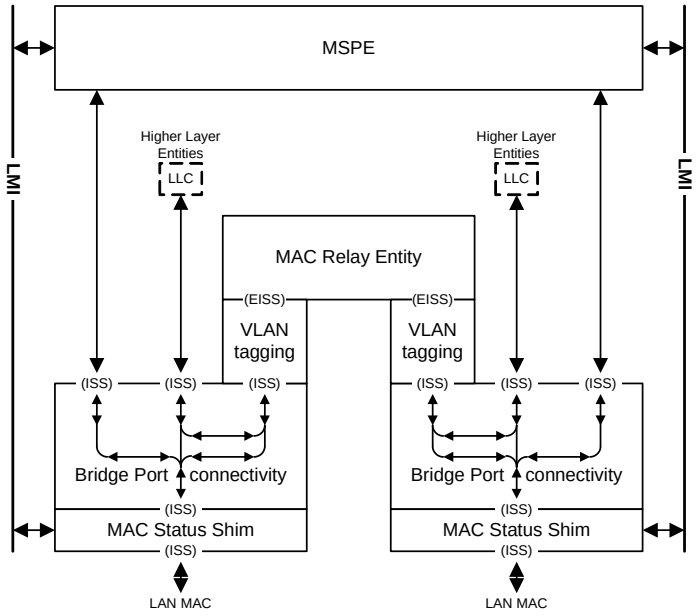


Figure 6—MAC Status Shims and the MAC Status Propagation Entity

7. State machines

Figure 7 shows a MAC Status Shim and its interaction with its peer(s) in the same system through the MSPE in more detail, including the necessary state machine variables.

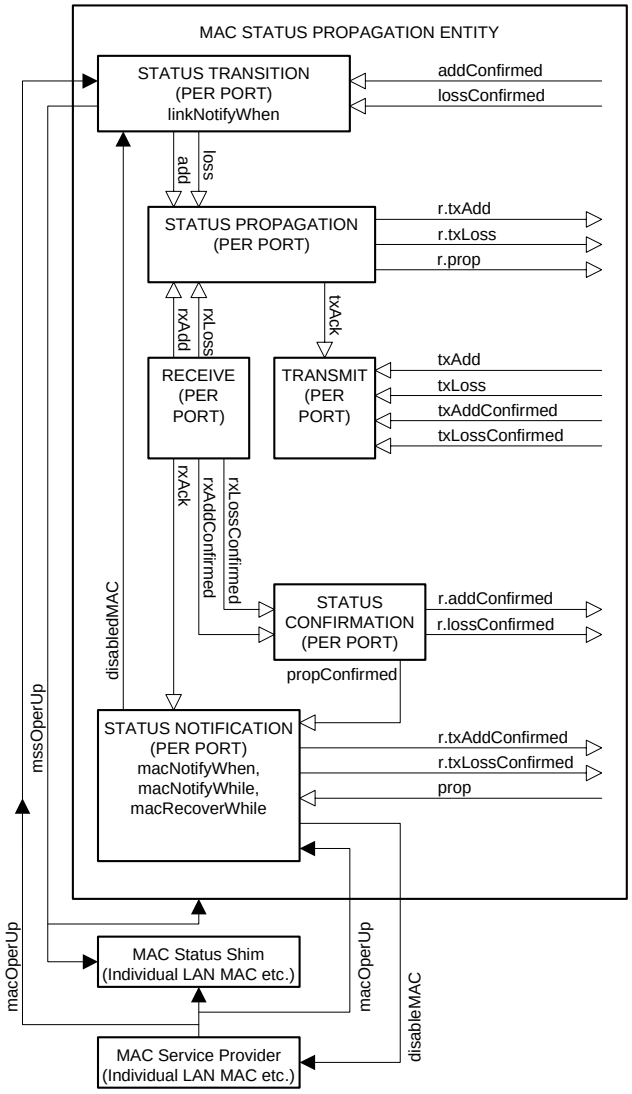


Figure 7—State machine overview

This note specifies the per port state machines that compose the MSPE—the Status Transition Machine (STM), Status Notification Machine (SNM), Status Propagation Machine (SPM), Status Confirmation Machine (SCM), and the supporting Transmit and Receive processes.

Some variables in Figure 7 are preceded with 'r.', this indicates that the variables referred to are those of corresponding state machines of the other port(s) of the relay. The operation of the state machines can be generalized, by the addition of explicit relaying elements, to provide MAC status propagation through a multi-port relay or to a relay that uses different status propagation protocols on some ports. However, the simplest application of the state machines of Figure 7 is to a relay with two ports, operating identical status propagation protocols on those ports. The MSPE state machines for such a relay comprise two sets of the per port state machines shown, with each of the variables preceded by 'r.' being that of the same name for the 'other' port.

The STM is responsible for telling the SPM about MAC_Operational transitions on its own Port (repeatedly if necessary), so that it can send link status notifications through the other Port(s).

The SNM is responsible for monitoring the progress of link status notifications sent through its Port (as notified by the SPM of the other Port), so that MAC status notification can be used (after a delay to allow the link status notification to elicit a confirmation) if necessary.

The SCM informs the other Port of receipt of a confirmation, as well as its own SNM.

7.1 State machine variables and timers

add — set by the STM to tell the SPM that the attached individual LAN has become operational, cleared by the SPM when it has taken note of that transition.

addConfirmed — set by the other Port's SCM to tell the STM that the addition has been confirmed, i.e. has been signaled one way or another to the system attached to the relay chain, or to a point where the relay chain ends (no further individual LAN, or LAN is OperDown), cleared by the STM when it has taken note of the confirmation.

disableMAC — provides communication from SNM to the individual LAN MAC (or other underlying service), via an LMI, resulting in some MAC specific action to disable the MAC in such a way that peer clients of the MAC are aware that it is disabled by inspecting own values of MAC_Operational for the MAC.

NOTE— The state machines do not assume that a client of the MAC can tell whether the MAC is not operational because that client has disabled it, or whether some other client has disabled it. The state machines also allow for the MAC to take some time to become operational after disabledMAC is set True.

disabledMAC — set by the SNM when it has intentionally disabled the underlying MAC, and for a recovery time following that, to communicate to the STM to ensure that blipping the MAC does not result in signaling a loss in the reverse direction.

loss — similar to add, but set to indicate that the attached individual LAN is no longer operational.

lossConfirmed — similar to addConfirmed but confirms a loss.

macOperUp — The value of MAC_Operational for the individual LAN MAC or other underlying service.

mssOperUp — the MAC_Operational signal to the MSS client.

prop — set by the other Port's SPM to notify the SNM that a change is being propagated through the Port.

propConfirmed — set by the SCM to notify the SNM that the change being propagated has been confirmed.

rxAck—set by the Receive process to tell the SNM that an acknowledgment has been received.

rxAdd—set by the Receive process to tell the SPM that an Add message is being propagated through the relay†1, cleared by the SPM.

†1If an instance of spanning tree is operating at the level of the relay (in addition to any spanning tree operating at the level of the systems connected to the periphery of the relay chain) additional considerations apply. These are only significant for multi-port relays and are not explored further in this note.

rxLoss— similar to add, but for a Loss.

txAck—set by the SPM to instruct the Transmit process to send an acknowledgment, cleared by the Transmit process.

txAdd—set by the other Port's SPM to causes transmission of an Add message through this Port, and cleared by the Transmit process.

txAddConfirm—set by the other Port's SNM to causes transmission of a message (through this Port) confirming that a received Add message has been acted upon. Cleared by the Transmit process.

txLoss—similar to a txAdd but causes a Loss message to be transmitted.

txLossConfirm—set by the other Port's SNM to causes transmission of a message (through this Port) confirming that a received Loss message has been acted upon. Cleared by the Transmit process.

7.2 State machine timers

Timers are implemented by variables that are decremented on each timer tick, with timer expiry occurring when they reach zero.

linkNotifyWhen — causes a link status notification to be sent on each expiry until the original status transition is confirmed.

macNotifyWhen — started when a change is first propagated through the Port (as signaled to the SNM by the MSPE), on expiry causes MAC status notification.

macNotifyWhile — sets the time for which the MAC is disabled for MAC status propagation.

macRecoverWhile — sets the time for which the mAC is permitted to be non-operational after being disabled before the link is reported as lost.

7.3 State machine procedures

No procedures are defined beyond those represented in the state machines.

7.4 State machines

Figure 8 shows the state machines, using the conventions common to 802.1 specifications.

7.5 Managing status propagation

In the absence of any better ideas and for the convenience of depicting timer values on the protocol time sequence diagrams in this note, the starting values of the timers have been given the following names:

linkNotifyWhen	: Tr — 'retry time'
macNotifyWhen	: Tw — 'wait time'
macNotifyWhile	: Td — 'down time'
macRecoverWhile	: Tm — 'recovery time'

The protocol behavior at each port can be managed by changing some of these times, as follows:

If Tr is set to zero, a single link status notification of loss or addition will be sent and a confirmation will not be required before the LAN is used by the MSS' client or another transition is signaled.

If Tw is set to zero, MAC status notification will be used immediately, without waiting for link status notification to work.

If Td is set to zero, then MAC status notification will not be used.

If both Td and Tw are zero then notification is skipped altogether, and the link status notification confirmed immediately.

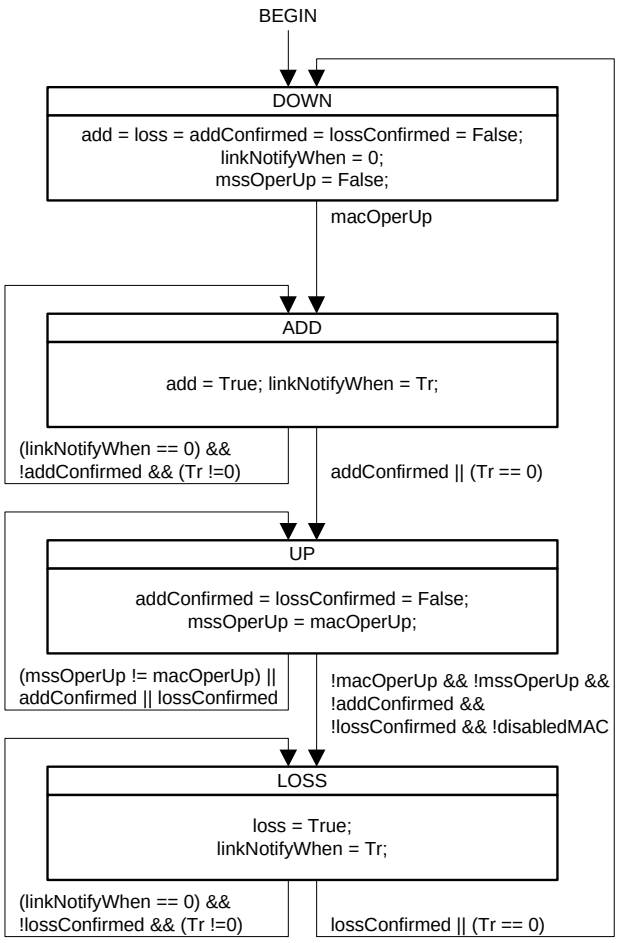
Clearly if we decide that all or any of Tr, Tw, and Td are fixed rather than manageable parameters, the same functionality can be provided by a flag for each, or by a mode switch that covers the useful combinations.

The default protocol behavior, i.e. when Tr, Tw, and Td are all non-zero, matches the time sequence diagrams above. When MAC_Operational transitions false on one port of a relay:

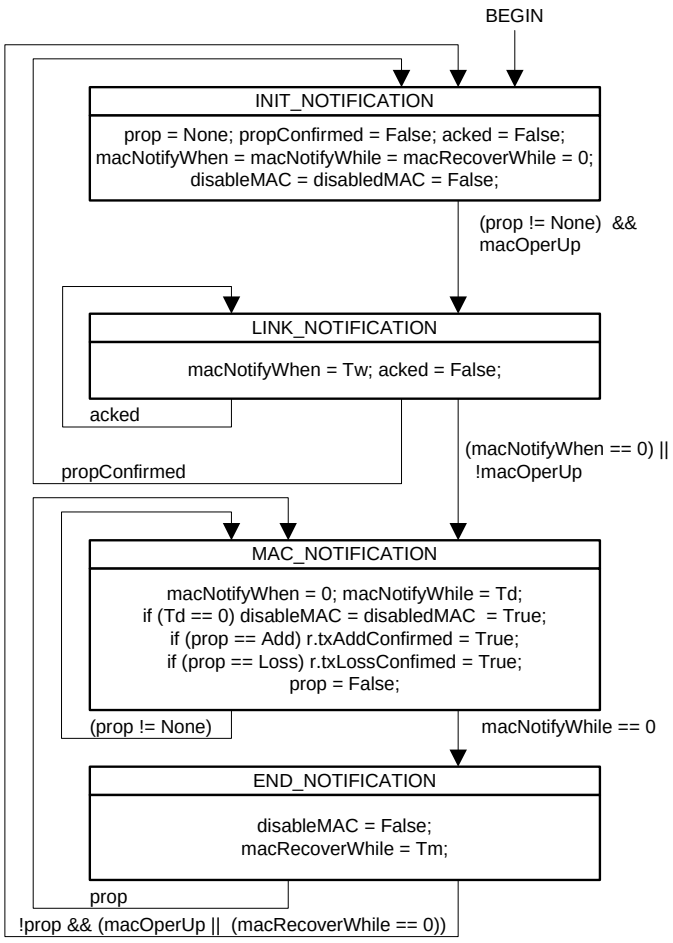
1) Status notification is initiated on the other port(s) with MAC_Operational true (if any)
2) Initially link status notification is used, and persists until an acknowledgment or confirmation is received or a timer expires
3) If the link status notification is unacknowledged by the time the timer expires, MAC status notification is used, again until an MAC status confirmation is received or a timer expires
4) The MAC status is allowed to revert true on the ports that are providing status notification.

Similarly, when MAC_Operational transitions true on one port of a relay:

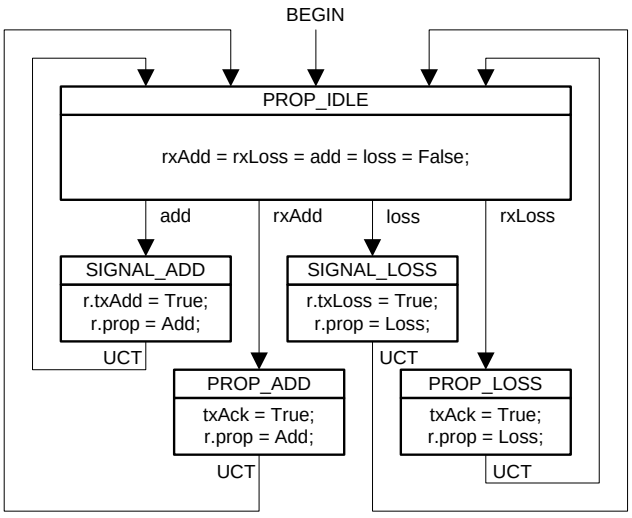
5) The value of MAC_Operational reported to the local client of the port is held false
6) Status notification is initiated on the other port(s) with MAC_Operational true, and proceeds as in (1) thru (4). If there are no such ports this step is skipped.
7) The MAC status (MAC_Operational) on the initiating port is allowed to become true



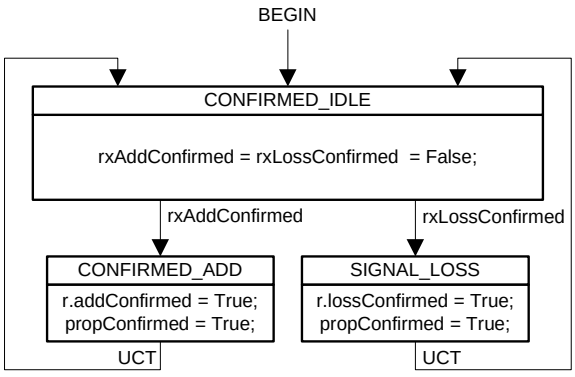
Status Transition Machine



Status Notification Machine



Status Propagation Machine



Status Confirmation Machine

Figure 8—State machines

8. Curious cases

Figure 4 (new connectivity) and Figure 5 (connectivity failure) above represent the target cases for the MAC sttaus propagation design. Before deciding that the or any proposed design is satisfactory we have to verify that it at least does no significant harm in cases where its help was not needed at all, and works in the (hopefully) rare cases of multiple near simultaneous transitions. Consideration of the latter should help in establishing formal goals for the protocol by revealing what happens, what is desirable, and what is achievable in complex cases.

Figure 10 shows recovery of the individual LAN at one end of at TPMR chain. The result is to ‘blip’ the MAC status at the other end of the chain, delaying availability of the recovered link by slightly more than Td. The figure also shows the effect of a possible effect of timing relationships between Tr and Td, as the initiator of the change rerties its Add message transmission just as the TPMR at the other end of the chain returns a confirm. The crossing of messages shown is unlikely, as the time sequence diagrams overemphasize the delay experienced by the Add and Confirm messages, but the effect is benign in any case.

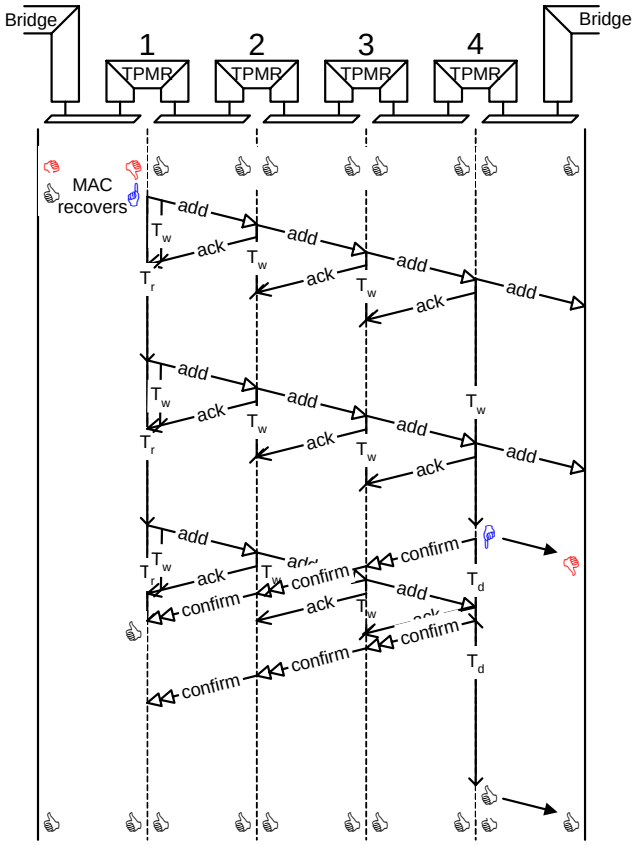


Figure 9—Recovery at end of chain

If the LAN between the end system and the first TPMR in the chain (1 above) had failed instead of recovering, the rest of Figure 10 would have looked the same, with the exception of the final state of that first LAN. The net effect of the status notification protocol is to transfer the change in mAC_Operational to the other end of the chain so that both ends see the transition. Clearly for best effects the transfer should complete in a short enough time so that the end system protocols at the initiating end of the link are still executing their initial state.

Figure 10 shows what happens if both ends of the link come up at the same time, the net effectis simply to delay the simultaneous start.

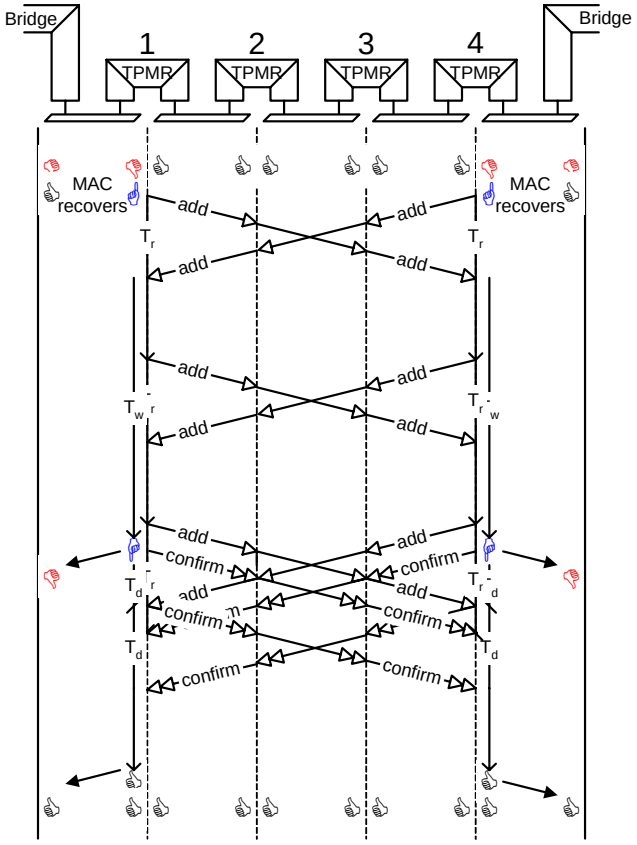


Figure 10—Near simultaneous transitions

Figure 11 shows simultaneous transitions of one LAN in the chain to OperUp and the other to OperDown, and illustrates a number of points about how the protocol or protocols do or should operate.

First, signalling of addition or loss can be somewhat arbitrary and depend on the relative timing of transitions on separate individual LANs. The same final connectivity can be represented by a loss followed by an addition, or an addition followed by a loss. If the implied resulting state is to mean anything then the information ‘connected as far as’, or ‘connected to and beyond’ also needs to be communicated. However, while this is a candidate for inclusion in any new protocol, support by existing protocols is unlikely. However this note persists with distinguishing Add and Loss messages, and their confirmations, because that ensures that closely spaced transitions, which might otherwise disguise unusual loss of higher layer protocol messages are not missed by the systems connected to the TPMR chain. Figure 12 shows a loss followed by recovery of the same LAN.

Second, no message is sent across the LAN between TPMRs 1 and 2 when the MAC is OperDown from the protocol clients point of view. That allows support of the protocol to be architected such that a message that signals ‘add(ition)’ or ‘loss’ is forwarded through the normal bridged path, so its speed of propagation does not depend on scheduling or notifying a status propagation process in each TPMR. This also means that we have a free choice as to what additional functionality can reside between the entities responsible for status propagation and the ‘real’ MAC†1.

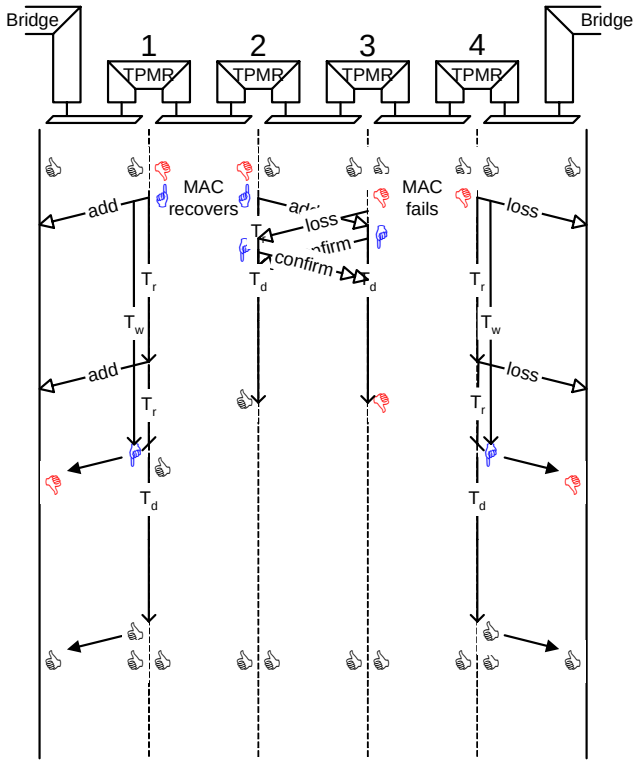


Figure 11—Near simultaneous transitions

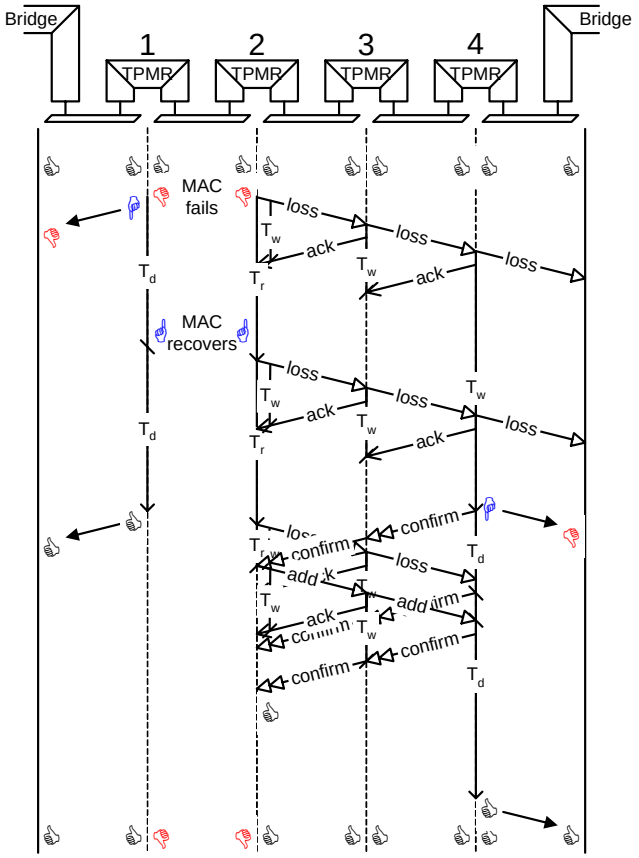


Figure 12—Loss with quick recovery

One last common case of simultaneous transitions is worth mentioning, what happens when the two ports of a TPMR power up or are enabled at the same time? Each tries to send a link notification through the other, and each link notification is discarded as th port is not yet available (mssOperUp false). The result is to ‘blip’ each LAN. The ends of the TPMR chain see an Add Loss Add sequence, if they are protocol capable), and a blip in MAC_Operational otherwise. If Tw is less than Td this will be a single blip.

9. Virtual links

The need for MAC status propagation arises from the standardization of TPMRs, relaying frames between LANs that are attached to higher layer bridges, but not participating in their spanning tree and other rapid configuration protocols. However the link notification signaling mechanisms described in this note could equally well be carried over multiplexed service instances, such as VLANs. MAC status notification could be used at the ends of those service instances, where a single (presumably untagged) VLAN is being carried over the MAC.

The multicast destination addresses of link notification frames clearly vary depending on the level of the link. In addition some relays along the link may wish to filter Add, Loss, or Confirm frames, as these relays are ‘repair’ points for the service being provided and are capable of switching connectivity to a spare link section (to give a part of a link a name).

[†1]Assuming such a thing exists any more.