10. Topology discovery and protection

Editors' Notes: To be removed prior to final publication.

References:
None.

Definitions:
None

Abbreviations:
None.

Revision History:

Draft 0.3, June 2002	Initial draft document for RPR WG review. Editorial notes added for clarification.
Draft 1.0, August 2002	Addressed comments from D0.3.
Draft 1.1, October 2002	Addressed comments from D1.0.
Draft 2.0, December 2002	Addressed comments from D1.1.
Draft 2.1, February 2003	Draft 2.1 for WG review, modified according to comments on D2.0.
Draft 2.2, April 2003	Draft 2.2 for WG ballot, modified according to comments on D2.1. Topology discovery and protection clauses combined.

Editors' Notes: *To be removed prior to final publication.*

The protection ad-hoc (PAH) is meeting to address the following list of issues:

1. Comment #691, #704, #21017: Generate and review state machine description for modification and validation of the topology database, including duplicate MAC address checks.

2. Comment #21017, March: Determine how to handle passthrough mode, including how to achieve fast topology discovery for passthrough when there are TP frames left on the ring from stations that have disappeared from the ring is an issue that must be resolved.

3. Comment #728, #758, #21011, March: Determine how to handle context containment.

4. Comment #764, March: Simplify protection state machine while maintaining the same behavior.

5. Comment #739, March: Is there a requirement for lockout?

10.1 Scope

This clause describes the RPR topology discovery and protection functions, which include the topology discovery protocol, the protection protocol, and the type length value (TLV) attribute discovery protocol. The topology discovery and protection protocol enables the fundamental RPR objectives of sub-50 millisecond resiliency and automatic, plug-and-play topology discovery. In addition, the rapid detection of topology changes is required to ensure that strict data frames can be delivered with no re-ordering or duplication. The TLV discovery protocol enables additional, less time-critical information to be reported by each station to the rest of the ring.

The protection protocol provides reliable mechanisms for sub-50 ms protection switching for all protected traffic on an RPR ring. It enables a mandatory protection mechanism called steering, and an optional protection mechanism called wrapping. This protocol ensures that each station receives link status change information, e.g., link failure or restoration information, required to make protection switching decisions reliably, and in the timeframe required. This protocol also ensures that stations steer or wrap in accordance

with the RPR protection hierarchy. This protocol resides in the MAC control sublayer, as shown in the shaded region of Figure 10.1.

The topology discovery protocol provides a reliable and accurate means for all stations on a ring to discover the topology of the stations on the ring. This includes both the initial topology and any changes to that topology. The topology discovery protocol also provides a mechanism for rapid detection of topology changes. This protocol resides in the MAC control sublayer, as shown in the shaded region of Figure 10.1.

The topology discovery protocol is closely related to the protection protocol in that both share the same control messaging mechanism. The information collected and used by both protocols is also logically stored together in a topology and status database. Information in the topology and status database is used by other protocols such as the RPR ringlet selection protocol, described in Clause 6, and the RPR fairness protocol, described in Clause 9.

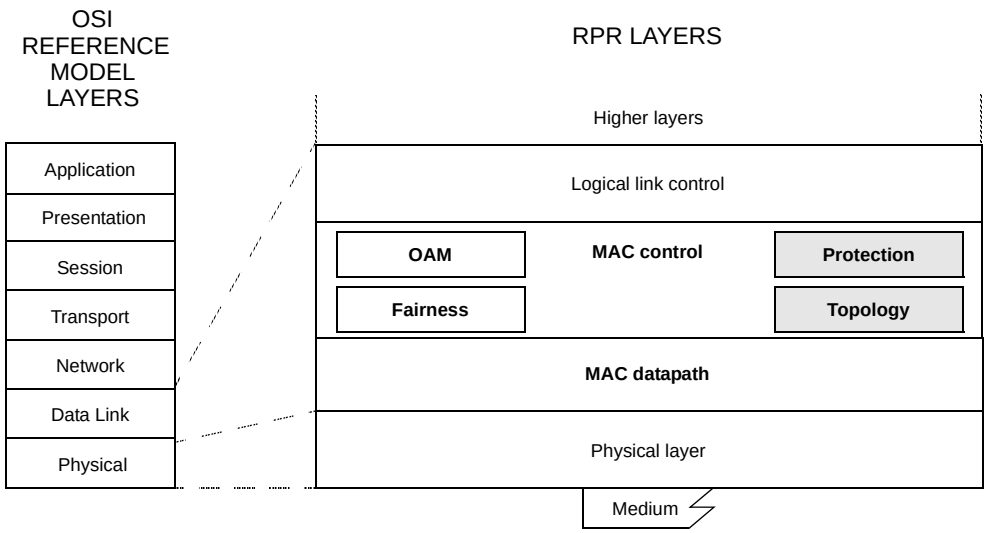


Figure 10.1—RPR layer diagram

The TLV discovery protocol provides a reliable and accurate means for all stations on a ring to discover additional, less time-critical information from the other stations on the ring. This protocol utilizes a distinct control messaging mechanism. The information collected and used by the TLV discovery protocol is also logically stored in the topology and status database.

All of these protocols are intended to be very scalable, to cause insignificant overhead for ring traffic, and to cause insignificant overhead on software and ASICs.

The services and features provided are:

a) Sub-50 ms protection switching for unicast and multicast traffic based on media “hard” or “soft” failures or based on operator invoked command.
b) Quick dissemination on the ring of information indicating media failures or operator invoked commands impacting use of the media.
c) Support of a standard protection hierarchy.
d) Greater bandwidth efficiency for unprotected services than path switching or bidirectional line switching.

e) Support of revertive and non-revertive protection switching operational modes.
f) Determination and validation of connectivity and ordering of stations on the ring.
g) Assurance that all stations on the ring converge to a uniform and current image of the topology.
h) Operation with all supported topologies: ring (closed loop), bus (broken ring or open loop), and isolated station.
i) Support of dynamic addition and removal of stations to/from the ring.
j) Scalability from one to hundreds of stations
k) Tolerance of frame loss.
l) Operation without any master station on the ring.
m) Operation independently of and in the absence of any management systems.
n) Validation of topology, including detection of mis-cabling between stations.
o) Means of sharing additional information between stations.
p) Support of the above services and features with insignificant overhead.

Editors' Notes: *To be removed prior to final publication.*

A complete discussion of topology discovery time will be added to the appropriate location in the clause when the mechanisms for topology discovery for passthrough scenarios are finalized.

Detecting mis-cabling (above) assumes static local assignment of ring IDs (as opposed to dynamic discovery). How the assignments are made is a local, non-standardized decision. This might need to be discussed in the OAM clause.

10.2 Overview

The RPR topology discovery protocol provides each station on the ring with knowledge of the number and arrangement of other stations on the ring. This collection of information is referred to as the topology image. Each station maintains its own local copy of the topology image for the entire ring in its topology database. Initially, the station's topology image contains information only about itself. The information required to create the basic topology image (including, for example, hop counts per ringlet from the local station to all other stations on the ring) is derivable from the topology and protection (TP) frames received from each station on the ring. TP frames are RPR control frames.

The link status, wrap status, and protection preference information required by the RPR protection protocol is also reported in the received TP frames. This information is also stored in the topology database for each station on the ring. The TP frame is used for this purpose because it is sent at a fast (substantially sub-50 millisecond) rate, as well as on triggers. A specification of the triggers for the initiation of topology discovery and the fast rate is provided in 10.9, along with a specification of the TP frame in 10.5. 10.4 also contains a description of the complete processing flow for received TP frames.

The protection protocol does not directly switch traffic on the data plane. Rather, it supplies essential information to entities within the MAC that do, such as ringlet selection, or directly controls when wrapping occurs. It utilizes link protection status information stored in the topology database to help determine if protection switching is required. It ensures that parameters such as hold-off time and wait to restore time specified by the network operator translate into correct station behavior. It ensures the robust interaction of operator-initiated protection requests and protection requests triggered by physical layer and MAC layer monitoring, within the framework of the RPR protection hierarchy.

The complete topology image contains more information than just station identifiers and physical connectivity relationships. It also contains station preferences information for use in various parts of this standard, and may contain optional vendor-specific information. Station preferences information is contained within the station TLV frame, while vendor-specific information is contained within the

vendor-specific TLV frame. TLV frames are RPR control frames. The information contained in the station and vendor-specific TLV frames is not as time-critical as that contained in the TP frame. A specification of the TLV frames is provided in 10.10, along with rules for the transmission and reception of TLV frames in 10.11.

The transmission of TP frames is initiated as needed and periodically. TLV frames are transmitted periodically. If the topology image is stable, the periodic TP and TLV frame transmissions will not result in any change to the topology database. Each station maintains its own topology database and follows a uniform set of rules for transmission and reception of TP frames and TLV frames. There is no master station.

The topology discovery protocol is robust for both fully connected bidirectional rings (loops) and for bidirectional rings in which edges are detected (chains). Edges are detected on spans through which data does not transit. For example, an edge exists at a station that is wrapping traffic. The complete definition of an edge is given in 10.7.

This clause also defines the modification and verification of the topology image upon receipt of TP frames or TLV frames. Topology stability and self-consistency must be validated because a failure of this verification is an indication of significant operational problems in the ring.

10.2.1 Initialization

At station initialization, the local topology image is initialized to contain only the local station and no links. Upon initialization, the station broadcasts TP frames on all ringlets following the TP frame transmission rules in 10.9.1. The station continually listens for TP frames broadcast on the ring, and updates its topology database upon receipt of TP frames from stations that are not contained in its topology database, or from stations already existing in its topology database for which TP frame contents have changed. Topology validation checks are executed to ensure that the topology becomes consistent within a period of time defined in 10.8.

On initialization the station is also triggered to broadcast station TLV frames (and if used, vendor-specific TLV frames) on all ringlets. After this initial trigger, the station broadcasts the TLV frames periodically, as defined in 10.11.1. Neighbor MAC address information contained in the station TLV frames is initialized to all 0's.

10.2.2 Addition of a station

When a station is inserted into an existing ring, it initializes itself as described above. Its neighbors detect the station as a new neighbor by receiving its TP frame with a *ttl* set to MAX_STATIONS (indicating the frame has traveled exactly one hop) and with the source MAC address set to a value other than that previously stored (if any).

The neighbor stations detect a new station on the ring, and respond by broadcasting a TP frame immediately on all ringlets. All other stations detect the new station by receiving its TP frame, and each responds by broadcasting a TP frame immediately on all ringlets. The information available from the TP frame includes the MAC address of the source station, the number of hops away the station is on the ringlet on which the frame is received (derived from *ttl*), and additional protection-related and time-sensitive station capabilities information described in 10.4.

All stations on the ring broadcast the TLV frames on all ringlets, and thus to all stations, based on the periodic timer at each local station. TLV frames are not triggered except by the periodic timer. Each TLV frame is partitioned into Type-Length-Value (TLV) entries, as defined in 10.10.3.

When one or more stations come out of passthrough mode, those stations behave as newly initialized stations. The behavior of the other stations on the ring is identical to that for stations detecting a newly initialized station on the ring.

Editors' Notes: *To be removed prior to final publication.*

The rules for determining a valid topology for passthrough scenarios when there are TP frames left on the ring from stations that have disappeared from the ring is an issue that will be resolved by the contribution to expand the contents of the topology validation subclause, 10.8.

10.2.3 Removal of a station

When a station is removed from a ring, the neighbor stations detecting the removal of the station take the protection actions described in 10.2.4. The neighbor stations detect a signal fail (SF) condition, and respond by broadcasting a TP frame immediately on all ringlets. All other stations detect this information by receiving the TP frames. The information available from the TP frame includes the MAC address of the source station, the number of hops away the station is on the ringlet on which the frame is received (derived from *ttl*), and additional protection-related and time-sensitive station capabilities information described in 10.4.

All stations on the ring broadcast the TLV frames on all ringlets, and thus to all stations, based on the periodic timer at each local station. TLV frames are not triggered except by the periodic timer. Each TLV frame is partitioned into Type-Length-Value (TLV) entries, as defined in 10.10.3.

When one or more stations go into passthrough mode, those stations disappear from the ring both from a control and data perspective. The stations adjacent to one or more stations that have gone into passthrough continue periodic transmissions of the TP frame. The first station that receives a TP frame from a different neighbor station than it saw previously is the first station to detect the passthrough event.

There is no objective or requirement to handle passthrough events in a 50 millisecond timeframe. 50 millisecond protection switching and service restoration is only applicable to media failures, e.g., ring and station failures resulting from a broken ring or a operator commanded protection event. The speed of passthrough detection currently depends on the setting of the TP frame transmission period.

Editors' Notes: *To be removed prior to final publication.*

There is a distinction between fast detection of passthrough and fast rediscovery of topology after a passthrough event that removes one or more stations from the ring. To discover passthrough quickly, a control messaging function is required between neighbor stations, and that control message needs to be sent periodically with a small period. To rediscover topology quickly after a passthrough event requires the triggering of broadcast TP frame transmissions by every station on the ring in response to detection of a passthrough event. The fast rediscovery of topology is not a requirement.

10.2.4 Protection event on span

A protection switching protocol that provides the services and features described in 10.1 must define:

a) A hierarchy of protection requests that requires local action by a station and reporting of the protection state to the rest of the ring. 10.6.2 includes description of the protection hierarchy, and of the definition of triggers resulting in specific protection requests.
b) The local actions that can be taken by stations on the ring to prevent traffic loss. As described in the rest of this subclause, these consist of steering or wrapping. The choice between steering and wrapping is dictated by the algorithm described in 10.6.1.

c) A mechanism and control frame format for reporting local protection state to the rest of the ring. This is done using the TP frame.
d) State machine and related requirements. These are given in 10.6. The state machine defines in detail what protection state information is reported to the rest of the ring, exactly when it is reported, and exactly when action is taken at a station to protect traffic.

When a span fails, the stations detecting the failure store the knowledge of their neighbor station address on the failed interface and clear their current knowledge of their neighbor station address on that interface. The stations send TP frames that report exactly which links failed to the other stations on the ring.

10.2.4.1 Steering protection

An overview of the behavior of a RPR steering ring is given in this subclause. If, for example, an equipment or facility failure is detected, the station(s) detecting the failure report the change in protection state to the rest of the RPR ring using the TP frame. When stations receive a TP frame indicating a failure, the topology database (defined in 10.4.4) at each station is updated accordingly. The topology database is used by ringlet selection (defined in 6.8.1) at each source station to direct its traffic onto ringlet0 or ringlet1 on a per destination station basis, whichever avoids the failed link.

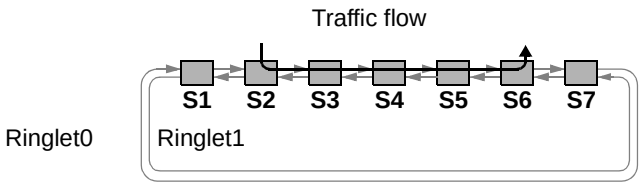


Figure 10.2—Data flow before fiber cut.

An example of the data paths taken before and after steering protection is shown in Figure 10.2 and Figure 10.3, respectively. Before the fiber cut, Station 2 sends to Station 6 via the path S2->S3->S4->S5->S6.

If there is a fiber cut between Station 3 and Station 4, Station 2 is first notified of the failure by TP frames sent by Station 3 and Station 4. Station 2 updates the appropriate fields in its topology database, then ringlet selection at Station 2 steers protected traffic destined for Station 6 onto ringlet1, using the path S2->S1->S7->S6.

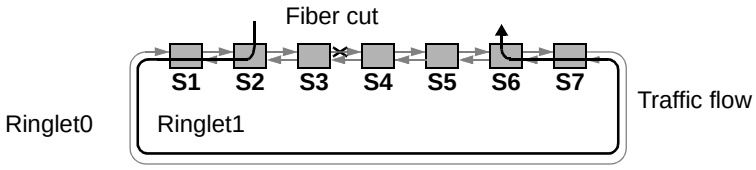


Figure 10.3—Data path after steering protection

Frames destined to a station beyond the point of failure that have been transmitted onto the ring before the steering database is updated at the source station are dropped at the failure point, since there is no delivery mechanism available.

For steering rings, when a link fails, multicast frames are sent in both directions, with the *ttl* set to the number of stations on the ring between the source station and the defective span on each direction. Duplicate strict mode frames must not be allowed to arrive at any station on the ring as a result of a protection condition ceasing to exist.

10.2.4.2 Wrap protection

An overview of the behavior of a RPR wrapping ring is given in this subclause. If an equipment or facility failure is detected, traffic going towards the failure is wrapped (looped) back to go in the opposite direction on the other ringlet (subject to the protection hierarchy). Wrapping takes place on the stations adjacent to the failure, under control of the protection switch protocol. The wrap re-routes the traffic away from the failure.

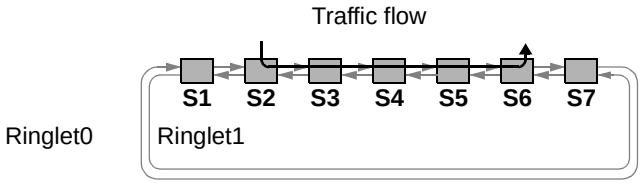


Figure 10.4—Data flow before fiber cut.

An example of the data paths taken before and after a wrap is shown in Figure 10.4 and Figure 10.5, respectively. Before the fiber cut, Station 2 sends to Station 6 via the path S2->S3->S4->S5->S6.

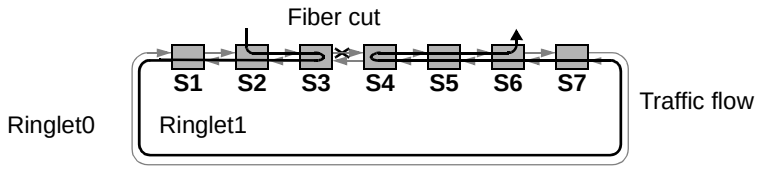


Figure 10.5—Data path after wrap

If there is a fiber cut between Station 3 and Station 4, Station 3 wraps traffic on ringlet0 to ringlet1, and Station 4 wraps traffic on ringlet1 to ringlet0. After the wraps have been set up, traffic from Station 2 to Station 6 initially goes through the non-optimal path S2->S3->S2->S1->S7->S6->S5->S4->S5->S6. S6 does not strip frames from the protection ringlet (ringlet1 in this case) to avoid frame mis-ordering during the protection event.

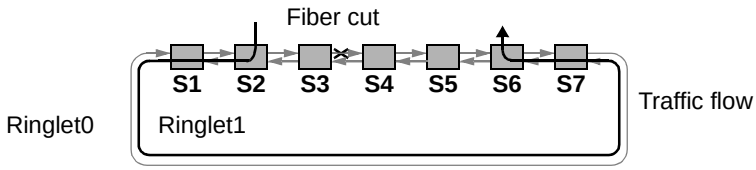


Figure 10.6—Data path after new topology discovery

Subsequently a new ring topology is discovered and a new optimal path S2->S1->S7->S6 may be used, as shown in Figure 10.6. Note that the topology discovery and the subsequent optimal path selection are not part of the protection protocol. The client, using the options defined in Clause 6, may choose to re-steer the traffic as shown in Figure 10.6, or may choose to keep the traffic wrapped, as shown in Figure 10.5

Editors' Notes: *To be removed prior to final publication.*

References will be fixed later based on where references appear (in Clause 2 or Appendix A).

The ring wrap is controlled through SONET BLSR style protection switch signaling. It is an objective to perform the wrapping at least as fast as specified in ANSI T1.105.01 for BLSR.

Editors' Notes: *To be removed prior to final publication.*

A comment will be filed on D2.2 with overview text for a topology validation subclause.

Editors' Notes: *To be removed prior to final publication.*

Proposed context containment scenarios will be posted for review before the May meeting. A comment will be filed on D2.2 with description and scenarios to for an overview subclause on context containment.

10.3 Variables and terminology used

Editors' Notes: *To be removed prior to final publication.*

The following items are specified in the MIB but are not yet included in this clause:

rprSpanProtectionCount
rprSpanProtectionDuration
rprSpanProtectionLastActivationTime

It may also be useful to separately report the protection status information reported by the PHY and that based on MAC layer mechanisms (RPR keepalives) for diagnostic purposes.

Additional variables need to be specified to cover topology validation checks and failure conditions.

Some of the variables used in the protection state machine are not yet included in this list, since it is expected that the protection state machine will be replaced in May with a modified, simpler state machine.

This clause defines the following terms and variables:

10.3.1 ***ringlet0Hops***: Calculated. The number of downstream hops on ringlet0. This is equal to the total number of stations on the ring in a loop topology, and is equal to the number of downstream hops until the edge span in a chain topology. This variable, along with *ringlet1Hops*, can be used to calculate the value used to set the MIB attribute *rprIfStationsOnRing*.

10.3.2 ***ringlet1Hops*****:** Calculated. The number of downstream hops on ringlet1. This is equal to the total number of stations on the ring in a loop topology, and is equal to the number of downstream hops until the edge span in a chain topology. This variable, along with *ringlet0Hops*, can be used to calculate the value used to set the MIB attribute *rprIfStationsOnRing*.

10.3.3 ***ringTopologyType*****:** Calculated. The type of topology, with possible values *LOOP* or *CHAIN*. The method used to calculate this is described in 10.7. This is used by the data path module. The value of this variable is used to set the MIB attribute TBD.

10.3.4 ***eastFormerNeighborMACAddress*****:** Calculated. The MAC address of the previously connected neighbor station on the east span. When there is a neighbor station connected on the east span, the value of this variable is the MAC address of the current neighbor station on the east span. The value of this variable is used to set the MIB attribute TBD.

10.3.5 ***westFormerNeighborMACAddress*****:** Calculated. The MAC address of the previously connected neighbor station on the west span. When there is a neighbor station connected on the west span, the value of this variable is the MAC address of the current neighbor station on the west span. The value of this variable is used to set the MIB attribute TBD.

10.3.6 ***stationMACAddress*****:** Calculated. The MAC address corresponding to a station entry in the logical topology database. There is one such entry for each station on the RPR ring. The value of this variable is used to set the MIB attribute *rprTopoImageMacAddress*.

10.3.7 *stationName*: Calculated. The station name field within a station entry in the logical topology database. There is one such entry for each station on the RPR ring. The value of this variable is used to set the MIB attribute *rprTopoImageStationName*.

10.3.8 *stationWrapPref*: Configured. This is mapped to the value of the *wp* (wrap protection preferred) bit in the TP frame sent by a station. The value of this variable is set by the MIB attribute *rprIfWrapPreferred*.

10.3.9 *stationJumboPref*: Configured. This is mapped to the value of the *jp* (jumbo frame preferred) bit in the TP frame sent by a station. The value of this variable is set by the MIB attribute *rprIfJumboFramePreferred*.

10.3.10 *ringProtectionType*: Calculated. The type of protection used on the ring. The method used to choose between steering and wrapping is described in 10.6.1. This is used by the data path module. The value of this variable is used to set the MIB attribute TBD.

10.3.11 *ringJumboType*: Calculated. The value of this variable indicates whether the ring supports reception of jumbo frames. The method used to determine this is described in 10.6.1. The value of this variable is used to set the MIB attribute TBD.

10.3.12 *eastSideProtectionState*: Calculated. The protection state on the east span within a station entry in the logical topology database. There is one such entry for each station on the RPR ring. The value of this variable is used to set the MIB attribute *rprTopoImageEastProtectionStatus*.

10.3.13 *westSideProtectionState*: Calculated. The protection state on the west span within a station entry in the logical topology database. There is one such entry for each station on the RPR ring. The value of this variable is used to set the MIB attribute *rprTopoImageWestProtectionStatus*.

10.3.14 *eastSideEdgeStatus*: Calculated. The edge status on the east span within a station entry in the logical topology database. The edge status is the wrap status for a station on a wrapping ring, and is a span not used for data transmission in a steering ring. There is one such entry for each station on the RPR ring. This is used by the data path module. The value of this variable is used to set the MIB attribute *rprTopoImageEastWrapStatus*.

10.3.15 *westSideEdgeStatus*: Calculated. The edge status on the west span within a station entry in the logical topology database. The edge status is the wrap status for a station on a wrapping ring, and is a span not used for data transmission in a steering ring. There is one such entry for each station on the RPR ring. This is used by the data path module. The value of this variable is used to set the MIB attribute *rprTopoImageWestWrapStatus*.

10.3.16 *ringlet0Reachability*: Calculated. The data reachability on ringlet0 from the local station to the station corresponding to the station entry in the logical topology database. There is one such entry for each station on the RPR ring. The method used to calculate this is described in 10.7. The value of this variable is used to set the MIB attribute *rprTopoImageRinglet0Reachability.*

10.3.17 *ringlet1Reachability*: Calculated. The data reachability on ringlet1 from the local station to the station corresponding to the station entry in the logical topology database. There is one such entry for each station on the RPR ring. The method used to calculate this is described in 10.7. The value of this variable is used to set the MIB attribute *rprTopoImageRinglet1Reachability.*

10.3.18 *ringlet0Weight*: Configured. This is mapped to the value of the *ringlet0Weight* field in the weight TLV in the station TLV frame sent by a station. The value of this variable is set by the MIB attribute *rprFairnessRingletWeight*. rprTopoImageEastWeight.

10.3.19 *ringlet1Weight*: Configured. This is mapped to the value of the *ringlet1Weight* field in the weight TLV in the station TLV frame sent by a station. The value of this variable is set by the MIB attribute *rprFairnessRingletWeight*. rprTopoImageWestWeight.

10.3.20 *ringlet0ResBW*: Configured. This is mapped to the value of the *ringlet0ReservedBW* field in the station bandwidth TLV in the station TLV frame sent by a station. The value of this variable is set by the MIB attribute *rprFairnessReservedRate*. rprTopoImageRinglet0ReservedRate.

10.3.21 *ringlet1ResBW*: Configured. This is mapped to the value of the *ringlet1ReservedBW* field in the station bandwidth TLV in the station TLV frame sent by a station. The value of this variable is set by the MIB attribute *rprFairnessReservedRate*. rprTopoImageRinglet1ReservedRate.

10.3.22 *eastNeighborMACAddress*: Calculated. The MAC address of the currently connected neighbor station on the east span as reported by the neighbor address TLV in the station TLV frame sent by a station. The value of this variable is used to set the MIB attribute TBD.

10.3.23 *westNeighborMACAddress*: Calculated. The MAC address of the currently connected neighbor station on the west span as reported by the neighbor address TLV in the station TLV frame sent by a station. The value of this variable is used to set the MIB attribute TBD.

10.3.24 *localRequest*: Configured. This is mapped to the value of *localRequest* used in the protection state machine. The value of this variable is set by the MIB attribute *rprSpanProtectionCommand*.

10.3.25 *IDLE*: Value. This protection state corresponds to the MIB setting RprProtectionStatus, noRequest.

10.3.26 *WTR*: Value. This protection state corresponds to the MIB setting RprProtectionStatus, waitToRestore.

10.3.27 *MS*: Value. This protection state corresponds to the MIB setting RprProtectionStatus, manualSwitch.

10.3.28 *SD*: Value. This protection state corresponds to the MIB setting RprProtectionStatus, signalDegraded.

10.3.29 *SF*: Value. This protection state corresponds to the MIB setting RprProtectionStatus, signalFailed.

10.3.30 *FS*: Value. This protection state corresponds to the MIB setting RprProtectionStatus, forcedSwitch.

10.3.31 *reversionMode*: Configured. This defines whether a station is operating in revertive or non-revertive mode. This is mapped to the value of *reversionMode* used in the protection state machine. The value of this variable is set by the MIB attribute *rprIfReversionMode*.

10.3.32 *wtrValue*: Configured. This is mapped to the value of the WTR timer period. The value of this variable is set by the MIB attribute rprIfProtectionWTR.

10.3.33 *fastTimerValue*: Configured. This is mapped to the value of the fast timer period for TP message transmissions. The value of this variable is set by the MIB attribute *rprIfProtectionFastTimer.*

10.3.34 *slowTimerValue*: Configured. This is mapped to the value of the slow timer period for TP message transmissions. The value of this variable is set by the MIB attribute *rprIfProtectionSlowTimer.*

10.3.35 *tlvTimerValue*: Configured. This is mapped to the value of the timer period for TLV message transmissions. The value of this variable is set by the MIB attribute TBD.

10.3.36 *holdoffTimerValue*: Configured. This is mapped to the value of the holdoff timer period for delaying the declaration of protection conditions. The value of this variable is set by the MIB attribute *rprSpanHoldOffTimer.*

10.3.37 *keepAliveTimeoutValue*: Configured. This is mapped to the value of the keepalive timeout period for declaration of a MAC layer keepalive failure. The value of this variable is set by the MIB attribute *rprIfKeepaliveTimeout*.

10.3.38 *totalRinglet0ReservedBW*: Calculated. The total reserved subclassA0 bandwidth on ringlet0. The method used to calculate this is described in 10.10.4.2. This is used by the fairness module, that needs this information to determine the total reclaimable bandwidth in 9.3.15. The value of this variable is used to set the MIB attribute TBD.

10.3.39 *totalRinglet1ReservedBW*: Calculated. The total reserved subclassA0 bandwidth on ringlet1. The method used to calculate this is described in 10.10.4.2. This is used by the fairness module, that needs this information to determine the total reclaimable bandwidth in 9.3.15. The value of this variable is used to set the MIB attribute TBD.

10.4 Flow chart for topology discovery and protection

A flow chart showing the overall flow of topology discovery and protection is shown in Figure 10.7. This flow chart represents the complete flow for a single station. The received frames mentioned in the flow chart are TP frames; the flow for TLV frames is described in 10.11. Detailed specification of the parts of this flow chart corresponding to normative state machines are given in various subclauses within Clause 10. An overview of the parts of this flow chart is given in the remainder of this subclause.

Editors' Notes: *To be removed prior to final publication.*

When additional information is added to this clause on topology validation and context containment, this subclause and the flow chart will be modified accordingly.

10.4.1 Local and external triggers

There is a protection state machine running on each side (east and west) of the local station. Protection-related events at the local station that cause entry to the protection state machine running on a given side of the station (referred to as this side, with the other side referred to as the other side) include:

a) Local operator administrative request on this side of the station.
b) Local operational failure on this receive interface of the station.
c) Clearing of a local administrative request or operational failure on this receive interface of the station.
d) WTR timeout on this receive interface of the station.
e) Administrative request or operational failure from the other interface of the station.

For example, for the protection state machine running on the west side of the station, this side refers to the west side and the other side refers to the east side.

The criteria for setting administrative request conditions, setting operational failure conditions, and clearing either type of condition are given in 10.6.2.

10.4.2 Receipt of topology and protection (TP) frames

Receipt of a TP frame from either ringlet first causes actions, including a sequence number check, described in 10.9.2. If the check passes, both of the protection state machines (running on the east and west sides of the station) are entered. The relevant information contained in the TP frame that is needed by the protection state machines is described in 10.9.2. If the check fails, the TP frame is discarded.

10.4.3 Protection state machine

The unified protection state machine for both steering and wrapping is defined in 10.6.3. There is one protection state machine running on each side of the station (east and west). The actions resulting from the protection state machine running on a given side of the station include:

a) Setting of the wrap status bit for that side of the station. This is used later in the flow chart to cause a station to wrap receive traffic on that side of the station.
b) Determination of the protection state to be set into the link protection state fields of the topology database for that receive interface of the station.
c) Determination of whether a TP frame is triggered due to a change in either the wrap status bit for that side of the station, or due to a change in the protection state for that receive interface of the station.

10.4.4 Topology database

There is a single central topology database that is modified based on the following:

a) Change in content of TP frames received from a particular station, or TP frame received from a new station.
b) Change in content of TLV frames received from a particular station, or TLV frame received from a new station.
c) Change of the protection state or wrap status for either side of the local station.

The definition of the topology database and the state machine for modifying and validating it are given in 10.4.4.

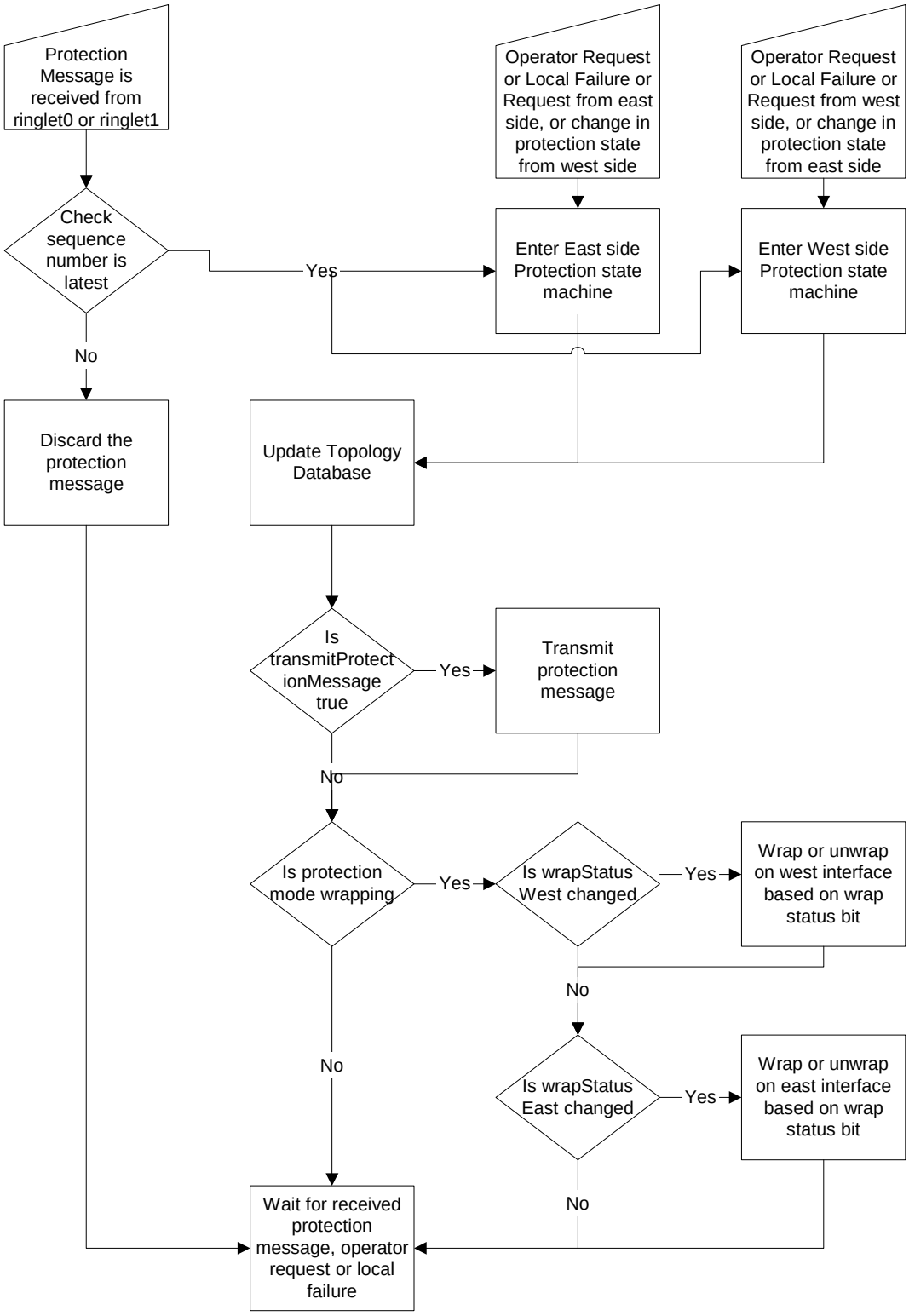


Figure 10.7—Flow chart for the protection protocol

Editors' Notes: *To be removed prior to final publication.*

Flow chart will be modified to FrameMaker format in an upcoming version of the draft.

10.4.5 Triggering of TP frame transmission

The state machine defining the triggers for TP frame transmission is given in 10.9.1.

10.4.6 Wrapping/steering action

Based on the setting of the wrap status bit for each side of the station and whether a ring is a wrapping ring, a wrap or unwrap on each side of the station is triggered at this point in the flow chart. For steering, the topology database has already been updated with modified link availability information, which provides the necessary information for ringlet selection to perform steering, as defined in 6.8.1.

10.5 Topology and protection (TP) frame format

The TP frame is a fixed length frame. The TP frame is used for signaling link protection state (defined in 10.6.2), for discovery of the physical topology of the ring, and for reporting of station preferences information. The information reported in the TP frame is time critical.

The TP frame format is outlined in Figure 10.8.

The header portion of Figure 10.8 corresponds to the RPR header of the RPR control frame defined in 8.3. The *controlVersion* and *controlType* fields correspond to the fields with those names that are part of the RPR payload of the RPR control frame.

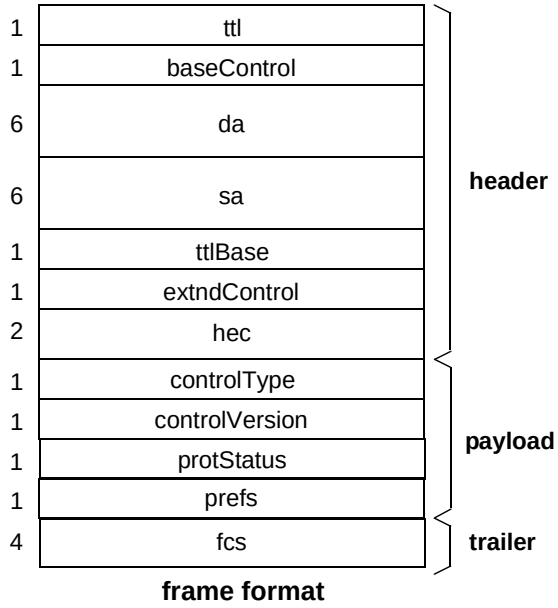


Figure 10.8—TP frame format

The TP frame is sent as a RPR control frame with a *controlType* value of CT_TOPO_PROT in Table 8.7. It is sent as a broadcast frame on all ringlets in order to minimize the transmission delay. It has a *ttl* (time to live) of MAX_STATIONS, is stripped by the source station, and has a *sa* (source address) set to the actual MAC of the sending station.

The *we* (wrap eligible) bit shall be set to zero for TP frames to ensure that those frames are not wrapped. The settings are described in 8.2.2.5.

10.5.1 *protStatus*

The 8-bit *protStatus* field is shown in Figure 10.9. The sub-fields of the *protStatus* field are described in 10.5.1.1 through 10.5.1.4.

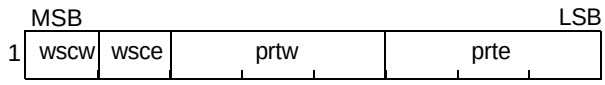


Figure 10.9—*protStatus* field format

10.5.1.1 *wscw*

The *wscw* (wrapping status code, west) bit is used on rings utilizing wrapping protection to indicate whether a wrap is present for traffic received on the west interface of a station. A value of 0 indicates that the interface is not wrapped, while a value of 1 indicates that the interface is wrapped. A value of 0 is used in steering rings.

10.5.1.2 *wsce*

The *wsce* (wrapping status code, east) bit is used on rings utilizing wrapping protection to indicate whether a wrap is present for traffic received on the east interface of a station. A value of 0 indicates that the interface is not wrapped, while a value of 1 indicates that the interface is wrapped. A value of 0 is used in steering rings.

10.5.1.3 *prtw*

The 3-bit *prtw* (protection request type, west) field is used to report the protection state on the west receive interface of a station.

The *prtw* values are defined in Table 10.1.

Table 10.1—*prtw and prte* values

Value (bin)	Name	Description
111_2	—	Reserved
110_2	—	Reserved
101_2	PRT_FS	Forced switch (FS)
100_2	PRT_SF	Signal fail (SF)
011_2	PRT_SD	Signal degrade (SD)
010_2	PRT_MS	Manual switch (MS)
001_2	PRT_WTR	Wait to restore (WTR)
000_2	PRT_IDLE	No request (IDLE)

10.5.1.4 *prte*

The 3-bit *prte* (protection request type, east) field is used to report the protection state on the east receive interface of a station.

The *prte* values are defined in Table 10.1.

10.5.2 *prefs*

The 8-bit *prefs* field is shown in Figure 10.10. The sub-fields of the *prefs* field are described in 10.5.2.1 through 10.5.2.3.

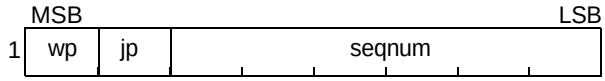


Figure 10.10—*prefs* field format

10.5.2.1 *wp*

The *wp* (wrap protection preferred) bit is used to indicate if a station prefers wrap protection. The rules for the selection of the protection mechanism to be used by an RPR ring is given in 10.6.1. A value of 0 indicates that wrap protection is not preferred, while a value of 1 indicates that wrap protection is preferred. A value of 0 is used by steering stations, and by wrapping-capable stations that are configured to prefer steering. The variable *stationWrapPref* contains the operator configured value used to set this field.

The *wp* field is included in the TP frame to minimize frame loss for scenarios where a steering station is inserted into a wrapping ring (see Annex K). Since TP frames are stripped by stations receiving frames on a wrapped span, the first TP frame sent by a newly inserted steering station is received by the neighboring stations (which are wrapped). Those stations immediately unwrap, but other stations on the ring do not find out that the ring has switched to steering mode until the next TP frame is transmitted by the steering station. To minimize frame loss, the *wp* field is included in the TP frame, which has the shortest fast transmission period.

10.5.2.2 *jp*

The *jp* (jumbo frame preferred) bit is used to indicate if a station supports jumbo frames. The rule for the determination of whether an RPR ring supports jumbo frames is given in 10.6.1. A more detailed description of what it means in RPR to support jumbo frames is given in 8.2. A value of 0 indicates that reception of jumbo frames is not preferred, while a value of 1 indicates that reception of jumbo frames is preferred. A value of 0 is used by stations that are not capable of receiving jumbo frames, and by stations capable of receiving jumbo frames that are configured to prefer not to receive jumbo frames. The variable *stationJumboPref* contains the operator configured value used to set this field.

The *jp* field is included in the TP frame so that station preferences information is reported in a single frame.

10.5.2.3 *seqnum*

The 6-bit *seqnum* (sequence number) field that is incremented each time that the contents of a TP frame change, e.g., the same sequence number is used for the all copies of a given TP frame until its contents change.

10.6 Protection protocol

10.6.1 Protection protocol rules

Editors' Notes: *To be removed prior to final publication.*

The rules below need to be reformatted to conform with the IEEE editorial template and style.

Fundamental protection rules in RPR include:

a) An RPR ring shall provide protection within 50ms of detection of a link or station failure.
b) All RPR stations shall provide support for steering, with support for wrapping optional.
c) All stations within the same RPR ring shall choose the same protection mechanism. Via the topology discovery protocol, every RPR station shall indicate if it prefers wrapping protection or not based on the configured local *stationWrapPref* setting. The indication in the TP frame is via the *wp* (wrap protection preferred) field defined in 10.5.2.1. If and only if all stations on an RPR ring advertise wrapping as the preferred protection mechanism, the wrapping protection mechanism shall be automatically used by all stations in the ring. Otherwise, steering shall be selected as the default protection scheme on the RPR ring.
d) A jumbo frame preferred station in an RPR ring shall behave as a non jumbo frame preferred station if any station on that ring reports that it is not jumbo frame preferred. This is via the *jp* (jumbo frame preferred) field defined in 10.5.2.2.
e) Revertive operation shall be the default mode of operation in RPR rings. Selection of revertive or non-revertive mode is supported through configuration of *reversionMode*. The behavior of non-revertive mode is equivalent to the behavior for an infinite WTR time. The behavior of a non-revertive station in terms of reporting and clearing of WTR shall be fully consistent with all rules pertaining to WTR.

10.6.1.1 TP frame transfer mechanism

a) TP frames are transferred in a broadcast frame format between stations on the ring. A received frame is passed to the receiving station's MAC control sublayer.
b) Triggering rules for TP frames are given in 10.9.1.
c) TP frames shall continue to be delivered on links that are in non-idle protection states.

10.6.1.2 RPR protection signaling mechanism

a) Protection switch signaling is performed using TP frames as defined in Figure 10.8.
b) A station transmits a TP frame on both ringlets. The definition of the transmission of TP frames is given in 10.9.

10.6.1.3 Additional protection protocol rules

a) The protection request hierarchy values are listed in Table 10.1 (listed from highest priority to lowest priority). In general, a higher priority request preempts a lower priority request within the ring, with exceptions noted as rules. The 3 bit values shown in the table correspond to the TP frame request type (PMRT) field in the TP frame.
b) When a station which initially detected a failure discovers the disappearance of the failure, it enters WTR (for a user-configured WTR time period *wtrValue*). The configurable range for WTR is defined in 10.6.2.
c) When a station is in WTR mode, and detects (via matching the MAC address of the new neighbor to the former neighbor MAC address on that span stored in the topology database, *eastFormerNeighborMACAddress* or *westFormerNeighborMACAddress*) that a new neighbor is not

the same as the old neighbor (stored while the old neighbor was still recognized as part of the topology), the station drops the WTR. This is done to allow a new station just connected into a ring to become data reachable from other stations as quickly as possible. The information from the topology database is needed so that protection knows when to apply WTR without requiring protection to separately store topology information.

d) When a station receives a local protection request of type SD or SF and it cannot be executed (according to protocol rules), it keeps the request pending. (The request can be kept pending outside of the protection protocol implementation.)

e) If a local non-failure request (WTR, MS, FS) clears, and if there are no other requests pending, the station enters idle state.

f) If there are two link failures and two resulting WTR conditions on a single segment, the second WTR to time out brings both the links up. (After a WTR time expires, a station does not unprotect automatically, but waits until it receives idle frames from its neighbor on the previously failed segment.)

g) The WTR on any link is dropped when a higher priority request is detected elsewhere on the ring. This requires the station monitoring the link with the WTR condition to check whether there are higher priority requests elsewhere on the ring.

h) A configurable hold-off time *holdoffTimerValue* must be supported per side of the station for protection triggers with a range of zero to at least 200 ms with 10 ms resolution. The default value is zero. The hold-off time is provided to prevent a spurious response to glitches on a link where such glitches are expected. For example, this could occur due to protection switching of RPR traffic by underlying SONET infrastructure.

i) An MS command on an interface of a station is rejected if there is another MS on the ring.

j) In case of the failure of a station's pass-through path (in Figure 6.4, between the ringlet0 datapath and the wrapW mux, or between the ringlet1 datapath and the wrapE mux), a station shall not wrap on both sides. An example is if a station includes two separate hardware modules (East and West) which are interconnected and the interconnecting pass-through cable or circuitry fails. A center wrapped station does not wrap on both sides because the two sides of the station would appear to be separate stations with duplicate MAC addresses. Rather, a center wrapped station shall choose to wrap on one of the two sides in case of failure in its pass-through path.

k) A TP frame shall be accepted only if the sequence number contained within the control header as defined in 10.5 meets the conditions defined in 10.9.2.

l) Requests higher than or equal to SF in the protection hierarchy can coexist. All requests above SF need to be cleared before the state is transferred into idle state.

m) Requests lower than SF in the protection hierarchy can not coexist with other requests. A higher priority request preempts a lower priority request.

For wrapping, two different spans with SF conditions result in simultaneous wraps on both spans. The desired behavior for two or more different spans with SD conditions is that the affected spans are equally usable, and therefore no wraps shall result. If two or more SD conditions, or two or more MS conditions, occur nearly simultaneously, then there may be a transient condition where there are wraps on some of the affected spans for a brief interval, but stations of these affected spans unwrap immediately when they find out that there is more than one SD condition, or more than one MS condition, in the ring. There is no corresponding condition for steering, as ringlet selection for traffic entering a ring with two or more spans with SD conditions can be handled independently at each source station. In this case ringlet selection treats all SD conditions equally when determining which direction on the ring to send traffic.

n) If a short path FS request is present on a given segment, and a SF condition takes place on the same segment, a station shall accept and process the SF condition ignoring the FS. (Without this rule, a single ended wrap condition could take place, wrapping on only one end of a segment.)

When a steering-preferred station is inserted into a wrapping ring, the station will send out TP frames on both ringlets indicating that it does not prefer wrapping. Its neighbor stations unwrap immediately upon finding out that there is a station on the ring that does not prefer wrapping. Subsequent TP frames are

forwarded to the rest of the ring. Upon receipt of these frames, all other stations on the ring proceed to steer their traffic.

When the only steering-preferred station is removed from a ring where all other stations prefer wrapping, its neighbor stations wrap immediately. Upon receipt of TP frames from the neighbor stations (and modification of the topology database) indicating that the only steering-preferred station has been removed from the ring, *ringProtectionType* is set to wrapping. This causes ringlet selection to steer traffic back to the original default direction for each destination station.

10.6.2 Protection hierarchy and triggers

The protection switch protocol processes the following request types (in the order of priority, from highest to lowest). The triggers for these requests are defined below. All requests are signaled using TP frames defined in 10.5.

1) Forced Switch (FS): Operator originated. Results in a protection switch away from a requested link (steering or wrapping all traffic away from the link). An example use of this request type is to add another station to the ring in a controlled fashion.
2) Signal Fail (SF): Automatic. Caused by a media Signal Failure or RPR keepalive failure. A media SF shall be detected based on the PHY_LINK_STATUS.indication defined in 7.2.3. Results in a protection switch away from the impacted link (steering or wrapping all traffic away from the link). SONET examples of SF triggers are: Loss of Signal (LOS), Loss of Frame (LOF), Line Bit Error Rate (BER) above a preselected SF threshold, and Line Alarm Indication Signal (AIS). Explicit definition of the SF triggers and SF clearing criteria for SONET are provided in the Telcordia GR-253-CORE and ANSI T1.105.01 standards, among others. An RPR keepalive failure is defined later in this clause. The configurable hold-off time *holdoffTimerValue*, if non-zero, starts after the physical SF condition is detected and shall elapse before SF triggers. Incorrect connection of interfaces of neighboring stations through miscabling shall also result in a MAC layer SF condition. Also, as described in Annex G, an internal failure within a station, such as the failure of a station's pass-through path, may be indicated as a SF condition on one side of the station to the rest of the ring.
3) Signal Degrade (SD): Automatic. Caused by a media Signal Degrade (e.g., excessive Bit Error Rate). A media SD shall be detected based on the PHY_LINK_STATUS.indication defined in 7.2.3. Results in a protection switch away from the impacted link (steering or wrapping all traffic away from the link). SONET examples of SD triggers are: Line BER or Path BER above a preselected SD threshold. Explicit definition of the SD triggers and SD clearing criteria for SONET are provided in the Telcordia GR-253-CORE and ANSI T1.105.01 standards, among others. The configurable hold-off time, if non-zero, starts after the physical SD condition is detected and shall elapse before SD triggers.
4) Manual Switch (MS): Operator originated. Like Forced Switch but of a lower priority. An example use of this request type is to force down a marginally operating link, but allow it to come back into use in the case of a more serious failure elsewhere on the ring.
5) Wait to Restore (WTR): Automatic. Entered after a link meets the restoration criteria following exit from a SF or SD condition. The protection switch protocol waits for the WTR time-out before restoring traffic. An example use of this request type is to prevent protection switch oscillations. The configurable range for WTR is defined later in this clause.

The protection module finds out about operator originated protection requests and clearing from the Layer Management Entity defined in Clause 12. It finds out about automatic protection requests via link status primitives defined in Annex B and Annex C for the Ethernet and SONET/SDH reconciliation sublayers, respectively. The protection module may also utilize automatic protection triggers mapped into SF or SD not explicitly defined in Annex B and Annex C.

Table 10.2 shows the mapping of LINK_STATUS values from the PHY to actual MAC protection state values, taking into account the internal MAC status. The internal MAC status may be Idle or SF, where SF is caused by an RPR keepalive failure or a ringlet identifier mismatch due to miscabling..

Table 10.2— Mapping of LINK_STATUS and internal MAC status to protection status

LINK_STATUS from PHY	Internal MAC status	MAC Protection Status
OK	IDLE	IDLE
	SF	SF
DEGRADE	IDLE	SD
	SF	SF
FAIL	IDLE	SF
	SF	SF
DEGRADE AND FAIL	IDLE	SF
	SF	SF

The purpose of RPR keepalives is to provide an indication that a station has an acceptable degree of operational capability. The transmission of RPR keepalives ideally should occur only if all implemented checks of normal MAC operation are fulfilled.

An RPR keepalive failure on a receive span is defined by the failure to receive any SC-FCM frames (as defined in Clause 9) for a configurable time range from 2 ms to 50 ms, with resolution 1 ms and a default value of 3 ms. This is set in *keepAliveTimeoutValue.* This means that to trigger an RPR keepalive failure, at least four consecutive SC-FCM frames must be missed, based on the 400 microsecond largest advertisement interval between SC-FCM frames.

A SF condition due to loss of keepalives is not cleared solely by the re-start of reception of keepalives from a newly connected or re-connected neighbor station. Upon receipt of a single keepalive, the link transitions to WTR state. For a new neighbor, if a TP frame is received from the neighbor before the WTR expires, the link is brought up upon receipt of the TP frame. If the WTR expires prior to receipt of a TP frame, then the link is brought up based on WTR regardless of whether the neighbor is a new neighbor or unchanged from before the SF condition.

In a passthrough mode scenario, the SC-FCM frame will be forwarded through the station or stations in passthrough. The detection of SF based on loss of keepalives is based on the same criteria at the receive station as if there were no stations in passthrough. The clearing criteria are also the same as if there were no stations in passthrough.

An SF condition due to miscabling is triggered by a ringlet ID mismatch on a TP frame received from a neighboring station (*ttl* == MAX_STATIONS).

The application of WTR upon clearing of an SF condition due to miscabling is the same as for any other SF condition. The SF condition is cleared by the receipt of a single TP frame received from a neighboring station with the correct ringlet ID. WTR is then applied based on whether the MAC address of the neighboring station is the same as or different from the MAC address of the previously connected neighboring station, as described in 10.6.1.

WTR shall be configured in *wtrValue* with values in the range of 0-1440 sec. with resolution of 1 second and with a default value of 10 seconds.

The following applies for both steering and wrapping rings unless stated otherwise.

Link failure or degradation indications (SF or SD) are reported to the entire ring, independent of whether there is a higher priority request in existence on the ring. The purpose of this reporting is to enable each station to have as complete a picture as possible of the status of all links on the ring, which is beneficial from a diagnostic perspective. This reporting is independent of whether, for wrapping rings, wrapping is caused by link degradation indications. A wrap shall always be initiated based on SF. A wrap shall be initiated based on SD only if there is no equal or higher priority request elsewhere on the ring, as detailed in 10.6.1.

A forced switch (FS) indication always goes into effect (and causes a wrap in wrapping rings) and is reported to the entire ring, except in the case described in (to prevent a single-ended wrap). A manual switch shall go into effect only if there is no equal or higher priority request elsewhere on the ring, as detailed in 10.6.1.

A wait to restore (WTR) is cleared immediately if a higher priority request is initiated from elsewhere on the ring.

Protection requests from any station travel around the ring and are stored in the topology and status database at all stations on the ring. For steering rings, to determine which ringlet is preferred for the transmission of frames from a given source to a given destination, the highest protection request on each path connecting the source and destination must be compared. For example, if there are links with protection request SD on the portion of ringlet0 connecting station A to station B, and there is a link with protection request SF on the portion of ringlet1 connecting station A to station B, then ringlet0 would be selected for steering traffic from station A to station B.

All protection switches are performed bidirectionally (protect at both ends of a segment for both transmit and receive directions, even if a failure is only unidirectional).

10.6.3 Protection state machine

This subclause specifies the unified state machine for both steering and wrapping protection. As described in 10.6.3.6, there is a protection state machine running on each span (east and west) of each station. State transitions that are specific to either steering or wrapping are noted within the state machine.

10.6.3.1 Inputs

thisSpanRequest
: Administrative state of the span of the station on which the protection state machine is running.
Values:
FS: Forced switch.
MS: Manual switch
IDLE: Idle.

thisSpanCurrentStatus
: Operational status of the span of the station on which the protection state machine is running.
Values:
SF: Signal fail.
SD: Signal degrade
IDLE: Idle.

rxTPFrame
: Received TP frame (may be from any station on the ring other than this station, and either ringlet). Relevant fields are *sa, ttl, ri, wscw, wsce, prtw, and prte*, as defined in 10.5.

reversionMode
Reversion mode configured for the station.
Values:
TRUE: Station is revertive.
FALSE: Station is non-revertive.

wtrExpired
Indication that wait to restore time on this span of the station has expired.
Values:
TRUE: Wait to restore time has expired.
FALSE: Wait to restore time has not yet expired, or wait to restore timer is not active.

wtrActive
Indication that wait to restore time on this span of the station is active.
Values:
TRUE: Wait to restore timer is active and has not yet expired.
FALSE: Wait to restore timer has expired, or is not active.

fromOtherSpanCurrentProtState
Protection state of the other span of this station. This is an input from the protection state machine running on the other span of this station.
Values:
FS: Forced switch.
SF: Signal fail.
SD: Signal degrade
MS: Manual switch
WTR: Wait to restore
IDLE: Idle.

10.6.3.2 Constants

FS
Forced switch.

SF
Signal fail.

SD
Signal degrade.

MS
Manual switch.

WTR
Wait to restore.

IDLE
Idle.

10.6.3.3 Variables

thisSpan
State of this span of this station. Relevant subfields are *admin*, *status,* and *ws*. *admin* indicates the administrative state of this span. *status* indicates the operational status of this span. *ws* indicates the wrap status of this span.

otherSideOfThisSpan
State of the other side of this span. Relevant subfields are *prt, sa,* and *ws. prt* indicates the protection state of this span. *sa* indicates the source address of the neighbor station on the other side of this span. *ws* indicates the wrap status of the other side of this span.

otherSpan
State of the other span of this station. Relevant subfield is *prt. prt* indicates the protection state of this span.

statusMod
This variable is TRUE if the input trigger to the state machine is a modified operational status of this span. It is FALSE otherwise.

adminMod
This variable is TRUE if the input trigger to the state machine is a modified administrative state of this span. It is FALSE otherwise.

transmitTPFrame
Indication that TP frames need to be triggered. This is an input to the TP frame transmission state machine defined in 10.9.1.

wrapCapable
Indication that a station is wrap capable.

nbrMACAddress
The most current source MAC address of the neighbor station on the other side of this span.

toOtherSpanCurrentProtState
Protection state of this span of this station at the output of the protection state machine. This is an input to the protection state machine running on the other span of this station, and corresponds to the input *fromOtherSpanCurrentProtState* in that state machine.

10.6.3.4 Functions

The functions below are used in the protection state diagram (or within functions used in the protection state diagram) presented later in this document.

AdminRequest
This function provides the value of a new administrative request, as set by *rprSpanProtectionCommand.*

TPFrameReceived
This function provides a newly received TP frame.

AdminClear
This function provides IDLE upon the clearing of an existing administrative request, as set by *rprSpanProtectionCommand.*

RejectRequest
This function rejects an administrative request.

StartWTR
The purpose of this function is to start the WTR timer.

```
StartWTR()
wtrActive = TRUE;
wtrExpired = FALSE;
```

ClearWTR
The purpose of this function is to clear the WTR timer.

ClearWTR()
wtrActive = FALSE;
wtrExpired = FALSE;

ProtOverriddenByNbr

The purpose of this function is to determine if the protection state of the receive link of this span connected to this station is overridden by the protection state of the other link of this span.

ProtOverriddenByNbr(thisSpanProt)
(thisSpanProt < SF) && (thisSpanProt < otherSideOfThisSpan.prt)
Values:
TRUE: The protection state of this span (either administrative state or operational status, depending on where this function is called from within the state machine) is overridden by protection on the other link of this span.
FALSE: The protection state of this span is not overridden.

ProtPreemptedByNonNbr

The purpose of this function is to determine if protection is preempted on this span by protection elsewhere on the ring, not including this span. The function NoOtherSpan is defined in 10.7.3.4. The variable *index* in the function below is -1 for the protection state machine running on the west span of the station, and 0 for the protection state machine running on the east side of the station.

ProtPreemptedByNonNbr(thisSpanProt)
(thisSpanProt < SF) && (((thisSpanProt <= rxTPFrame.prt) && !TPFrameFromNbr()) || NoOtherSpan(index, thisSpanProt))
Values:
TRUE: The protection state of this span (either administrative state or operational status, depending on where this function is called from within the state machine) is preempted by protection elsewhere on the ring, not including this span.
FALSE: The protection state of this span is not preempted.

NewNbrDetect

The purpose of this function is to determine if a new neighbor station is connected to this station on this span as defined in Figure 10.15.

NewNbrDetect()
(otherSideOfThisSpan.sa != nbrMACAddress)
Values:
TRUE: There is a new neighbor station connected to this station on this span.
FALSE: There is not a new neighbor station connected to this station this span.

TPFrameFromNbrOnShortPath

The purpose of this function is, for a given received TP frame *rxTPFrame*, to determine if the frame comes from the neighbor station on the short path (as defined in Figure 10.15). The constant RINGLET in the function below is RINGLET_0 for the protection state machine running on the west span of the station, and RINGLET_1 for the protection state machine running on the east side of the station.

TPFrameFromNbrOnShortPath()
(rxTPFrame.ttl == MAX_STATIONS) &&
(rxTPFrame.ri == RINGLET)
Values:

TRUE: Frame comes from neighbor station on short path.
FALSE: Frame does not come from neighbor station on short path.

TPFrameFromNbr

The purpose of this function is, for a given received TP frame *rxTPFrame*, to determine if the frame comes from the neighbor station (on either the short path or the long path as defined in Figure 10.15). The constant RINGLET in the function below is RINGLET_0 for the protection state machine running on the west span of the station, and RINGLET_1 for the protection state machine running on the east side of the station.

```
TPFrameFromNbrOnShortPath()
((rxTPFrame.ttl == MAX_STATIONS) &&
 (rxTPFrame.ri == RINGLET)) || ((rxTPFrame.ri == OTHER_RINGLET)
 && (rxTPFrame.sa == nbrMACAddress))
```

Values:
TRUE: Frame comes from neighbor station.
FALSE: Frame does not come from neighbor station.

SetProtectionInfo

The purpose of this function is to set the variables used to update the topology database to the appropriate values. This function is written for a protection state machine running on the west side of this station. The identical function runs on the east side of the station, with west replaced by east.

```
SetProtectionInfo()
Info.westSpanProtectionState = max(thisSpan.status, thisSpan.admin);
Info.westSpanOpStatus = thisSpan.status;
Info.westSpanAdminState = thisSpan.admin;
Info.westWrapStatus = thisSpan.ws;
Info.westNeighborMACAddress = otherSideOfThisSpan.sa;
```

10.6.3.5 State encodings

The operational status codes used in the protection state machine are described in Table 10.3.

Table 10.3— Link *status* field values

Value	Name	Description
n/a	IDLE	None or an insignificant error rate
	SD	A significant (but acceptable) error rate
	SF	An unacceptable error rate

The administrative directive codes used in the protection state machine are described in Table 10.4

The values placed within the *prtw* and *prte* fields in the transmitted TP frame are derived from the aforementioned *status* and *admin* fields, as described in Table 10.5.

Table 10.4— Administrative *admin* field values

Value	Name	Description
n/a	IDLE	Normal operation, desire to activate the span
	MS	Abnormal operation, desire to deactivate the span
	FS	Abnormal operation, insistance on deactivating the span

Table 10.5—Encoded TP frame protection field values

admin	*status*	*WtrActive*	*prtw*/*prte* field	Description
FS	any	any	PRT_FS	Highest directive
below FS	SF	—	PRT_SF	Higher condition
below FS	SD	—	PRT_SD	Lower condition
MS	—	any	PRT_MS	Lowest directive
IDLE	IDLE	1	PRT_WTR	Wait to restore
		0	PRT_IDLE	Normal operation

10.6.3.6 Protection state table

Editors' Notes: *To be removed prior to final publication.*

The state machine does not yet cover the scenario where a steering preferred station is either inserted or removed from a ring containing all wrapping preferred stations. Currently the portions of the state machine that are not required for stations that are not wrap capable are marked.

The current sequence number check has the problem that it does not take the ringlet identifier of the received TP frame into account. Therefore, if a TP frame is lost from the neighbor station on the short path and the TP frame with the same contents is received on the long path, the subsequent TP frame on the short path is not accepted for processing.

The state machine in Table 10.6 shows the required conditions for the following types of protection actions to be taken:

a) Setting of the wrap status bit for a span of the station. This is used later in the flow chart shown in Figure 10.7 to cause a station to wrap.
b) Determination of the protection request value to be set into the protection state fields of the topology database for each span of the station.
c) Determination of whether a TP frame is triggered due to a change in either the wrap status bit for a span of the station, or due to a change in the protection request value for a receive link of the station.

The initial state is the START state. Optional rows are shaded in grey. In the case of any ambiguity between the text and the state machine, the state machine shall take precedence. The notation used in the state machine is described in 3.4.

There is a protection state machine running for each span of the station. The state machine in Table 10.6 is written generically to correspond to either span of the station.

Protection-related events at the local station that cause entry to the protection state machine include:

a) Administrative request on this span of the station.
b) Operational protection status change on this span of the station.
c) Clearing of an administrative request on this span of the station.
d) WTR expiration on this span of the station. (Non-revertive mode is handled by not starting the WTR timer.)
e) Administrative request or operational protection status change from other span of the station. This is the sole dependence of the protection state machine running on one span of the station on the protection state machine running on the other span of the station.

In addition, a received TP frame from the ring results in entry to the protection state machine.

It is important to note that upon each entry to the protection state machine, the variables thisSpan, otherSideOfThisSpan, and otherSpan are initially set to values contained in the topology database. The correspondence is as follows:

a) For the protection state machine running on the west span of the station: thisSpan.prt = topoDB[0].westSideProtectionState, thisSpan.status = topoDB[0].westSideOpStatus, thisSpan.admin = topoDB[0].westSideAdminState. For the protection state machine running on the east span of the station: thisSpan.prt = topoDB[0].eastSideProtectionState, thisSpan.status = topoDB[0].eastSideOpStatus, thisSpan.admin = topoDB[0].eastSideAdminState.
b) For the protection state machine running on the west span of the station: otherSideOfThisSpan.prt = topoDB[1].eastSideProtectionState. For the protection state machine running on the east span of the station: otherSideOfThisSpan.prt = topoDB[-1].westSideProtectionState.
c) For the protection state machine running on the west span of the station: otherSpan.prt = topoDB[0].eastSideProtectionState. For the protection state machine running on the east span of the station: otherSpan.prt = topoDB[0].westSideProtectionState.

.

Table 10.6—Protection state machine for one span of a station

Current state		Row	Next state	
state	**condition**		**action**	**state**
START	thisSpanRequest=AdminRequest(), thisSpanRequest!=NULL	1	—	REACT

	thisSpan.status != thisSpanCurrentStatus	2	thisSpan.status=thisSpanCurrentStatus;
			statusMod=TRUE;
			if (wtrActive)
			ClearWTR();
				PROCESS

	rxTPFrame=TPFrameReceived(),(rxTPFrame!=NULL) && TPFrameFromNbrOnShortPath()	3	otherSideOfThisSpan.prt=rxTPFrame.prt;
			nbrMACAddress=rxTPFrame.sa;
			
			otherSideOfThisSpan.ws=rxTPFrame.ws;
			(wrap capable stations only)	

	thisSpanRequest=AdminClear(), (thisSpanRequest!=NULL)	4	thisSpan.admin=thisSpanRequest; adminMod=TRUE;	
	otherSpan.prt != fromOtherSpanCurrentProtState	5	otherSpan.prt=fromOtherSpanCurrentProtState;	
	wtrExpired || !TPFrameFromNbr()	6	—	
	—	7	—	

Table 10.6—Protection state machine for one span of a station *(continued)*

Current state		Row	Next state	
state	**condition**		**action**	**state**
PROCESS	thisSpan.admin==FS	8	—	FINISH

	(thisSpan.status==SF) && (otherSideOfThisSpan.prt==FS)	9	thisSpan.ws= 1;
			(wrap capable stations only)
			
			thisSpan.admin=IDLE;
			otherSideOfThisSpan.prt=IDLE;
			if (statusMod)
			transmitProtectionFrame=TRUE;	

	thisSpan.status==SF	10	thisSpan.ws=1;
			(wrap capable stations only)
			
			thisSpan.admin=IDLE;
			if (statusMod)
			transmitProtectionFrame=TRUE;	

	(thisSpan.status==SD) && ProtOverriddenByNbr(thisSpan.status)	11	thisSpan.ws=1;
			(wrap capable stations only)
			
			thisSpan.admin=IDLE;
			if (statusMod)
			transmitProtectionFrame=TRUE;	

	(thisSpan.status==SD) && ProtPreemptedByNonNbr(thisSpan.status)	12	thisSpan.ws=0;
			(wrap capable stations only)
			
			thisSpan.admin=IDLE;
			if (statusMod)
			transmitProtectionFrame=TRUE;	

	thisSpan.status==SD	13	thisSpan.ws=1;
			(wrap capable stations only)
			
			thisSpan.admin=IDLE;
			if (statusMod)
			transmitProtectionFrame=TRUE;	

	thisSpan.admin==MS && ProtOverriddenByNbr(thisSpan.admin)	14	thisSpan.ws=1;
			(wrap capable stations only)
			
			thisSpan.admin=IDLE;
			transmitProtectionFrame=TRUE;	

	thisSpan.admin==MS && ProtPreemptedByNonNbr(thisSpan.admin)	15	thisSpan.ws=0;
			(wrap capable stations only)
			
			thisSpan.admin=IDLE;
			transmitProtectionFrame=TRUE;	

	thisSpan.status==MS	16	—	

	adminMod==TRUE	17	thisSpan.ws=0;
			(wrap capable stations only)
			
			transmitProtectionFrame=TRUE;	

Table 10.6—Protection state machine for one span of a station *(continued)*

Current state		Row	Next state	
state	**condition**		**action**	**state**

PROCESS	ProtPreemptedByNonNbr(WTR)	18	if (wtrActive)
			ClearWTR();
			
			thisSpan.ws=0;
			(wrap capable stations only)
			
			if (statusMod)
			transmitProtectionFrame=TRUE;
				FINISH

	ProtOverriddenByNbr(WTR)	19	if (wtrActive)
			ClearWTR();
			if (statusMod)
			transmitProtectionFrame=TRUE;	

	(!wrapCapable() && wtrExpired) || (wrapCapable && wtrExpired && (otherSideOfThisSpan.prt==WTR))	20	ClearWTR();
			thisSpan.status=IDLE;
			transmitProtectionFrame=TRUE;	

	wrapCapable && wtrExpired && TPFrameFromNbrOnShortPath () && (otherSideOfThisSpan.prt!=WTR)	21	ClearWTR();
			thisSpan.status=IDLE;
			thisSpan.ws=0;
			transmitProtectionFrame=TRUE;	

	wtrActive	22	if (!reversionMode) ClearWTR();	

	statusMod && !NewNbrDetect()	23	if (reversionMode)
			StartWTR();
			thisSpan.status=WTR;
			transmitProtectionFrame=TRUE;	

	NewNbrDetect()	24	ClearWTR();
			thisSpan.status=IDLE;
			otherSideOfThisSpan.sa=nbrMACAddress;
			
			thisSpan.ws=0;
			(wrap capable stations only)
			
			transmitProtectionFrame=TRUE;	

	wrapCapable && !ProtPreemptedByNonNbr(otherSideOfThisSpan.prt)	25	thisSpan.ws=1; transmitProtectionFrame=TRUE;	

	wrapCapable	26	if (thisSpan.ws==1)
			transmitProtectionFrame=TRUE;
			thisSpan.ws=0;	

	!wrapCapable	27	—	

Table 10.6—Protection state machine for one span of a station *(continued)*

Current state		Row	Next state	
state	**condition**		**action**	**state**

FINISH	—	28	statusMod=FALSE;
			adminMod=FALSE;
			thisSpanRequest=NULL;
			rxTPFrame=NULL;
			toOtherSpanCurrentProtState=max(thisSpan.admin, thisSpan.status);
			SetProtectionInfo();
				START

REACT	(thisSpanRequest==MS) && ProtOverriddenByNbr(thisSpanRequest)	29	RejectRequest();	START
	(thisSpanRequest==MS) && ProtPreemptedByNonNbr(thisSpanRequest)	30	RejectRequest();	

	—	31	thisSpan.admin=thisSpanRequest;
			
			thisSpan.ws=1;
			(wrap capable stations only)
			
			transmitProtectionFrame=TRUE;
			adminMod=TRUE;
			if (wtrActive)
			  ClearWTR();	

Row 10.6-1: Administrative request received on this span of the station.

Row 10.6-2: Operational status of receive link of this span of the station has changed. Set status of this span into the appropriate variable. Note that an IDLE reported by the underlying physical or MAC layer link status detection will be converted to WTR when appropriate later in this state machine.

Row 10.6-3: If a TP frame is received from the neighbor station directly connected on this span, extract the relevant information from this frame for use later in the state machine. This information includes protection state, wrap status, and source address.

Row 10.6-4: Administrative request cleared on this span of the station. Initialize variables used later in the state machine with appropriate information from the topology database.

Row 10.6-5: Protection state of receive link of other span of the station has changed. Set status of other span into the appropriate variable.

Row 10.6-6: WTR timer has expired, or a TP frame is received from the neighbor station on this span.

Row 10.6-7: Otherwise transition directly to PROCESS state.

Row 10.6-8: No additional action required when administrative state of this span is FS. Actions upon FS command are executed in the REACT state.

Row 10.6-9: Operational status of this span is SF, and administrative state of the other side of this span is FS. This results in a wrap (if required), and in the clearing of any non-idle administrative state on this side of

this span. If the operational status of the span has changed, then transmitTPFrame is set to TRUE for the state machine in Table 10.9. The FS on the other side of this span is ignored in this case, since there is an SF on this side of this span.

Row 10.6-10: Operational status of this span is SF. This results in a wrap (if required), and in the clearing of any non-idle administrative state on this side of this span. If the operational status of the span has changed, then transmitTPFrame is set to TRUE for the state machine in Table 10.9.

Row 10.6-11: Operational status of this span is SD, and protection switching due to this condition is overridden by a higher priority protection state on the other side of this span. This results in a wrap (if required), and in the clearing of any non-idle administrative state on this side of this span. If the operational status of the span has changed, then transmitTPFrame is set to TRUE for the state machine in Table 10.9.

Row 10.6-12: Operational status of this span is SD, and protection switching due to this condition is preempted by a higher priority protection state any ring span other than this span. This results in no wrap (if required), and in the clearing of any non-idle administrative state on this side of this span. If the operational status of the span has changed, then transmitTPFrame is set to TRUE for the state machine in Table 10.9.

Row 10.6-13: Operational status of this span is SD, and protection switching due to this condition is not overridden or preempted. This results in a wrap (if required), and in the clearing of any non-idle administrative state on this side of this span. If the operational status of the span has changed, then transmitTPFrame is set to TRUE for the state machine in Table 10.9.

Row 10.6-14: Administrative state of this span is MS, and protection switching due to this condition is overridden by a higher priority protection state on the other side of this span. This results in a wrap (if required), and in the clearing of any non-idle administrative state on this side of this span. TransmitTPFrame is set to TRUE for the state machine in Table 10.9.

Row 10.6-15: Administrative state of this span is MS, and protection switching due to this condition is preempted by a higher priority protection state any ring span other than this span. This results in no wrap (if required), and in the clearing of any non-idle administrative state on this side of this span. TransmitTPFrame is set to TRUE for the state machine in Table 10.9.

Row 10.6-16: Administrative state of this span is MS, and protection switching due to this condition is not overridden or preempted. No additional action is required in this case. Actions upon new MS command are executed in the REACT state.

Row 10.6-17: Administrative state of this span is cleared to IDLE (and the operational status of this span is also IDLE). This results in no wrap (if required), and transmitTPFrame is set to TRUE for the state machine in Table 10.9.

Row 10.6-18: A protection state of WTR or higher in the hierarchy is present on a ring span other than this span. This results in the clearing of WTR if the WTR timer is active, and the removal of any wrap on this span (if required). If the operational status of the span has changed, then transmitTPFrame is set to TRUE for the state machine in Table 10.9.

Row 10.6-19: A protection state of MS or higer in the hierarchy is present on the other side of this span. This results in the clearing of WTR if the WTR timer is active. If the operational status of the span has changed, then transmitTPFrame is set to TRUE for the state machine in Table 10.9.

Row 10.6-20: If a station is not wrap capable and WTR has expired, then WTR is cleared, the status of this span is set to IDLE, and transmitTPFrame is set to TRUE for the state machine in Table 10.9. The same actions are taken if the station is wrap capable, WTR has expired, and the protection state of the other side of this span is WTR.

Row 10.6-21: Optional, wrap capable station only. If WTR has expired, the protection state on the other side of the span is not equal to WTR, and this information comes from a TP frame received on the short path from the neighbor station, then WTR is cleared, the span is unwrapped, the status of this span is set to IDLE, and transmitTPFrame is set to TRUE for the state machine in Table 10.9. The same actions are taken if th

Row 10.6-22: If the wait to restore timer is active and the station is in non-revertive mode, then clear the WTR. Otherwise do nothing.

Row 10.6-23: If the status of this span has changed and a new neighbor station has not been detected on this span, then set the status of this span to WTR. TransmitTPFrame is set to TRUE for the state machine in Table 10.9. If the station is in revertive mode, start the WTR timer.

Row 10.6-24: If a new neighbor station is detected on this span, clear the WTR timer, set the status of this span to IDLE, set the source address for the other side of this span to the source MAC address of the new neighbor station, and unwrap (if required). TransmitTPFrame is set to TRUE for the state machine in Table 10.9.

Row 10.6-25: Optional, wrap capable station only. Protection on this span triggered by the protection state on the other side of this span is not preempted by the protection state on any other ring span. This results in a wrap on this span, and transmitTPFrame is set to TRUE for the state machine in Table 10.9.

Row 10.6-26: Optional, wrap capable station only. Protection on this span triggered by the protection state on the other side of this span is preempted by the protection state on any other ring span. This results in no wrap on this span. TransmitTPFrame is set to TRUE for the state machine in Table 10.9 if the span was previously wrapped.

Row 10.6-27: If the station is not wrap capable, do nothing.

Row 10.6-28: The purpose of this state is to clear variables for the next pass through this state machine, to set the current protection state variable to be used by the state machine running on the other span of this station, and to set the protection information into the variables used to set this information into the topology database.

Row 10.6-29: If an administrative request of MS is overridden by the protection state on the other side of this span, then the request is rejected.

Row 10.6-30: If an administrative request of MS is preempted by the protection state of any ring span other than this span, then the request is rejected.

Row 10.6-31: If the administrative request is MS and is not overridden or preempted, or if the adminstrative request is FS, then the request is accepted. This results in setting the administrative state of this span, a wrap (if required), and the clearing of the WTR timer if it is active. TransmitTPFrame is set to TRUE for the state machine in Table 10.9.

10.7 Topology database

Under certain conditions such as a failed span in a bidirectional ring, a station is only able to receive TP and TLV frames from the other stations on the ring on one ringlet. It is a requirement to be able to construct a complete topology database reflecting connectivity information on both ringlets. Hence, a single topology database showing information for both ringlets can meet this requirement as well as facilitate maintenance of a consistent database. The hop count to another station can be calculated through examination of the topology image. For each ringlet, the topology image must be completed up to the station whose hop count is desired.

There are multiple representations of the topology database given in this subclause. The first is a logical view of the database that conforms to the MIB-level view. The second is a lower level view of the database that can be modified easily for various topology perturbations.

10.7.1 Logical representation of topology database

There is one entry (row) for each station known to be on the ring. The values of each entry (not including the TLV entries covered in 10.10.1) are:

a) the MAC address of the station described by the entry, *stationMACAddress,*
b) the downstream hop count on ringlet0, *ringlet0Hops,*
c) the downstream hop count on ringlet1, *ringlet1Hops,*
d) station preferences, *stationWrapPref* and *stationJumboPref,*
e) the link protection state of the receive link from the west station, *westSideProtectionState,*
f) the link protection state of the receive link from the east station, *eastSideProtectionState,*
g) the edge or wrap status of the station's west span, *westSideEdgeStatus,*
h) the edge or wrap status of the station's east span, *eastSideEdgeStatus,*
i) the data reachability downstream on ringlet0, *ringlet0Reachability,* and
j) the data reachability downstream on ringlet1, *ringlet1Reachability.*

The receive link protection state can be any of the TP frame request values given in Table 10.1. The reserved subclassA0 bandwidth is described in 10.10.4.2.

An example table for both ringlets is shown in Table 10.7.

Table 10.7—Topology and status database table

local MAC	hop count for ringlet0	hop count for ringlet1	station prefer-ences	west span protec-tion state	east span protec-tion state	west span edge or wrap status	east span edge or wrap status	data reach-ability, ringlet0	data reach-ability, ringlet1	TLV fields
00-10-A4-97-A8-DE	0	0	JP=0 WP=1	IDLE	IDLE	No	No	Yes	Yes	see 10.10.1
00-10-A4-97-A8-EF	1	3	JP=1 WP=1	IDLE	IDLE	No	No	Yes	Yes	see 10.10.1
00-10-A4-97-A8-AC	2	2	JP=1 WP=1	IDLE	IDLE	No	No	Yes	Yes	see 10.10.1
00-10-A4-97-A8-BD	3	1	JP=1 WP=0	IDLE	IDLE	No	No	Yes	Yes	see 10.10.1

The information used to configure values in the topology database come from a variety of sources. As described in 10.7.2, received TP frames are used to update station MAC address and hop count information on each ringlet. Protection state information (wrap status and link availability) and station capabilities information are also carried within received TP frames. Edge status for steering rings is calculated at the

receive station. Data reachability is derived as described in 10.7.2. The remainder of the information in the database is obtained from TLV frames

The topology database is modified upon receipt of TP or TLV frames that result in a change in information contained in the database, or based on input from the protection state machine. Changes in parameters reported within TLV frames or by the protection state machine result in the modification of fields within the database, but not the position of stations in the topology.

There is also non-per-station information stored in the logical topology database. From the information in the topology database, a number of useful pieces of information can be derived. These are shown in Table 10.8. For the topology database in Table 10.7, the corresponding ringlet0 and ringlet1 edge status information both correspond to LOOP (for fully connected bidirectional ring), rather than to CHAIN (for a string). The number of destination stations on ringlet0 or on ringlet1 is 3 for this case.

Table 10.8—Ring level topology and protection information

ring topology type	ring protection type	ringlet0 destination stations	ringlet1 destination stations	former west neighbor's MAC	former east neighbor's MAC
LOOP	NOT WRAPPING	3	3	00-10-A4-97 -A8-EF	00-10-A4-97 -A8-BD

The ring level fields are:

a) ring topology type, *ringTopologyType*,
b) ring protection type, *ringProtectionType*,
c) number of destination stations on ringlet0, *ringlet0Hops*,
d) number of destination stations on ringlet1, *ringlet1Hops*,
e) former west neighbor station's MAC address, *westFormerNeighborMACAddress*, and
f) former east neighbor station's MAC address, *eastFormerNeighborMACAddress*.

The method of calculating the variables *westSideEdgeStatus*, *eastSideEdgeStatus*, *ringlet0Reachability*, *ringlet1Reachability*, *ringTopologyType*, *ringlet0Hops*, and *ringlet1Hops* are described in 10.7.2.

A station determines itself to be isolated if the receive links on both the east and west interfaces of a station have a protection state of signal fail (SF). An isolated station shall remove all entries from its local topology image, but shall transfer its own east neighbor and west neighbor information to the former east neighbor and former west neighbor fields in the database before clearing the east neighbor and west neighbor fields. The information in the former east neighbor and former west neighbor fields is needed to determine whether a link should be brought up immediately, as described in 10.6.1.

The topology database also contains the MAC address information of the neighbor previously connected to each interface of the local station. This information is only relevant to the local station, and can be stored separately from the information shown in Table 10.7. This information is stored in the topology database because it is required by the protection protocol. The topology module notifies the protection module when this information has changed. The former neighbor entries are rewritten upon a change in the value of the current neighbor entry for each respective interface.

Upon detection of signal fail on a given interface (west for example), the west neighbor MAC address is invalidated as described in 10.7.2 (cleared from the logical topology database point of view). Upon detection of a new neighbor on that interface, the MAC address of the new neighbor is first compared to the former west neighbor MAC address. The protection module is notified if the new neighbor and former neighbor MAC addresses are different. The new neighbor MAC address is then copied into both the west neighbor MAC address and the former west neighbor MAC address.

10.7.2 Lower level representation of topology database

The MAC has topology database components illustrated in Figure 10.11. The signed index associated with this database represents the ringlet0 distance-to-source hops. The positive-index and negative-index entries are derived from protection frames received on ringlet1 and ringlet0 respectively; the magnitude of the index equals *256* minus the protection frame's *ttl* value.

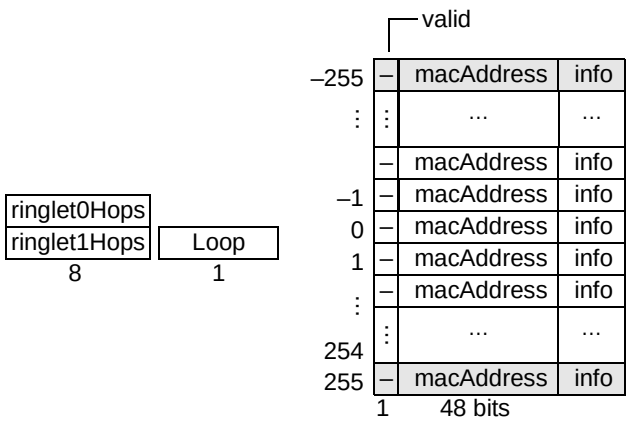


Figure 10.11—Topology database parameters

The *ringlet0Hops* and *ringlet1Hops* fields specify numbers of ringlet0 and ringlet1 destination stations respectively. The *Loop* bit is 0 and 1 for chain and loop topologies respectively.

Within each array, the *valid* bit values of 0 and 1 correspond to unvalidated and validated entry status respectively. The *valid* bit is cleared for all stations downstream of a detected edge; the *valid* bit is set when a protection frame is received from a station beyond a detected edge in a chain topology. The 48-bit *macAddress* field and the *info* field are copies of fields within protection frames.

Since the database components are sometimes accessed at line-rate speeds, each datapath is expected to have a distinct copy of the topology database. This avoids mandating a high-speed communication path between datapaths, at a modest cost of supporting low-rate datapath-to-datapath discovery-frame communications.

To calculate *westSideEdgeStatus* and *eastSideEdgeStatus*:

In a steering ring, an edge is detected based on the following rules:

An edge is present on a span if:

a) FS or SF present on one or both links of any span, or
b) SD present on one or both links of span, and there is no other span with SD condition, or
c) MS present on one or both links of span, and there is no other span with MS condition, or
d) WTR present on one or both links of span, and there is no other span with WTR condition.

In a wrapping ring, an edge is present wherever there is a wrap. The above check for steering rings matches the conditions that will result in at least one wrap being present on the ring. The implication of having an edge present on a given span is described under station validity.

A half-severed link transmits information in one direction but not the other (only one of the two links is operational). Half-severed links, just like fully severed links are edges in that data does not flow across an edge link from a frame transfer perspective.

To determine validity of an entry in the topology database:

In a wrapping ring, station validity is cleared for stations downstream on a ringlet from a wrap. Station validity is set for a station downstream on a ringlet when a protection message is received from that station from upstream on the other ringlet, except for TP frames received across a wrapped interface.

In a steering ring, station validity is cleared for stations downstream on a ringlet from a detected edge. Station validity is set for a station downstream on a ringlet when a TP frame is received from that station from upstream on the other ringlet, except for TP frames that have traversed an edge span.

When a topology is detected to be a loop, all entries in the topology database outside of the loop are cleared.

To calculate *ringlet0Reachability* and *ringlet1Reachability*:

Data reachability in a wrapping ring is always set to yes for all stations within the valid topology.

Data reachability in a steering ring is set as follows. A destination station is marked data reachable from a source station on a given ringlet if there is no edge span on the path from source station to destination station on that ringlet. A prerequisite for making this decision is that all stations on the path from source to destination on a ringlet be known in the topology and marked valid. A destination station is marked non data reachable from a source station on a given ringlet if there is an edge span on the path from source station to destination station on that ringlet.

The purpose of explicitly showing downstream data reachability in the topology database is to provide a clear indication of stations that are downstream data reachable from a given station, while at the same time enabling the complete physical connectivity of the ring topology to be shown.

To calculate *ringTopologyType*:

In a wrapping ring, this is set to CHAIN if there is a wrap (edge) present anywhere on the ring. This is set to LOOP if there is no wrap present anywhere on ring. In the event that two chains have been connected into a loop, this check will pass only when the full topology has been discovered and all stations in the topology are marked as valid.

In a steering ring, this is set to CHAIN if there is an edge present anywhere on the ring. This is set to LOOP if there is no edge present anywhere on ring. In the event that two chains have been connected into a loop, this check will pass only when the full topology has been discovered and all stations in the topology are marked as valid.

To calculate *ringlet0Hops* and *ringlet1Hops*:

In a wrapping ring, these variables are set based on downstream hop count distance to the nearest span on which there is a wrap. The ringlet0 hop count is the number of downstream hops on ringlet0 to a station in wrap. The ringlet1 hop count is the number of downstream hops on ringlet1 to a station in wrap. If the topology is a closed loop, then ringlet0 hop count = ringlet1 hop count = number of stations in ring - 1.

In a steering ring, these variables are set based on downstream hop count distance to the nearest span on which the edge check above is satisfied. The ringlet0 hop count is the number of downstream hops on ringlet0 to an edge span. The ringlet1 hop count is the number of downstream hops on ringlet1 to an edge span. If the topology is a closed loop, then ringlet0 hop count = ringlet1 hop count = number of stations in ring - 1.

Editors' Notes: *To be removed prior to final publication.*

Whether these rules need to be modified in passthrough scenarios must be evaluated.

10.7.2.1 Stable database structures

Stable database structures could reflect loop or chain topologies, as illustrated in the left and right sides of Figure 10.12 respectively. In a loop topology, the local station's entries are repeated on the two ends of the loop. This facilitates verification that the topology is a loop. In a chain topology, each side of the chain terminates where an edge is detected.

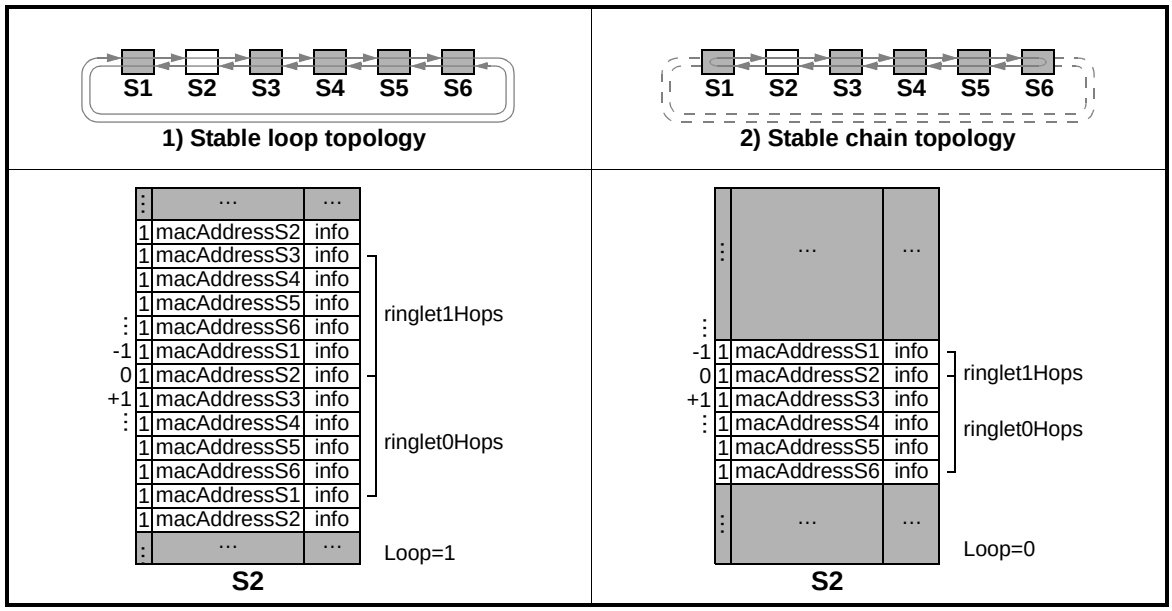


Figure 10.12—Stable database structures

Editors' Notes: *To be removed prior to final publication.*

Additional work needs to be done by the PAH on this subclause, including the addition of a flow chart/state machine. The PAH will target submission of modified text prior to the next meeting.

Editors' Notes: *To be removed prior to final publication.*

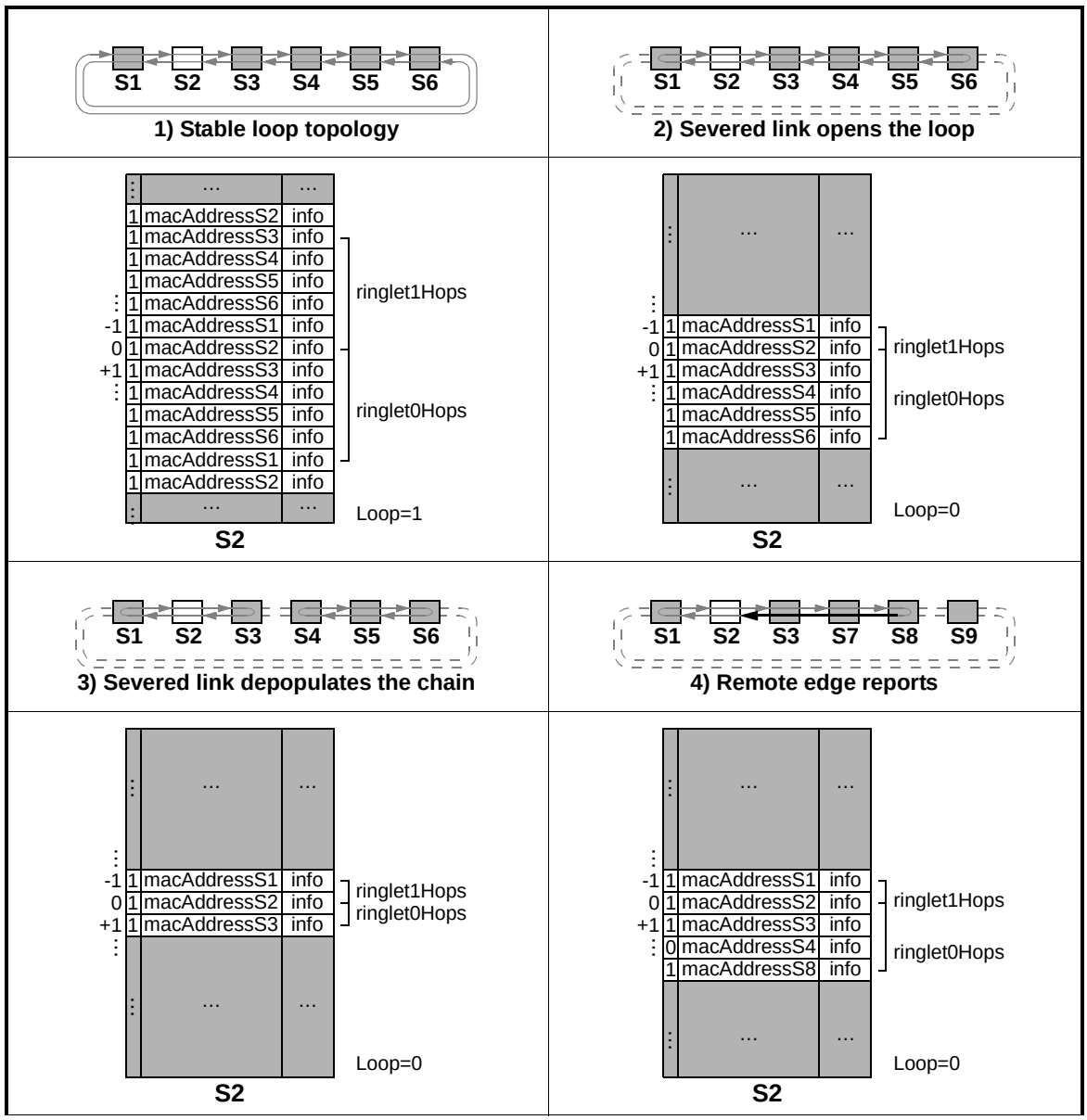
The tables masquerading as Figures 10.12 and 10.13 must be converted into figures.

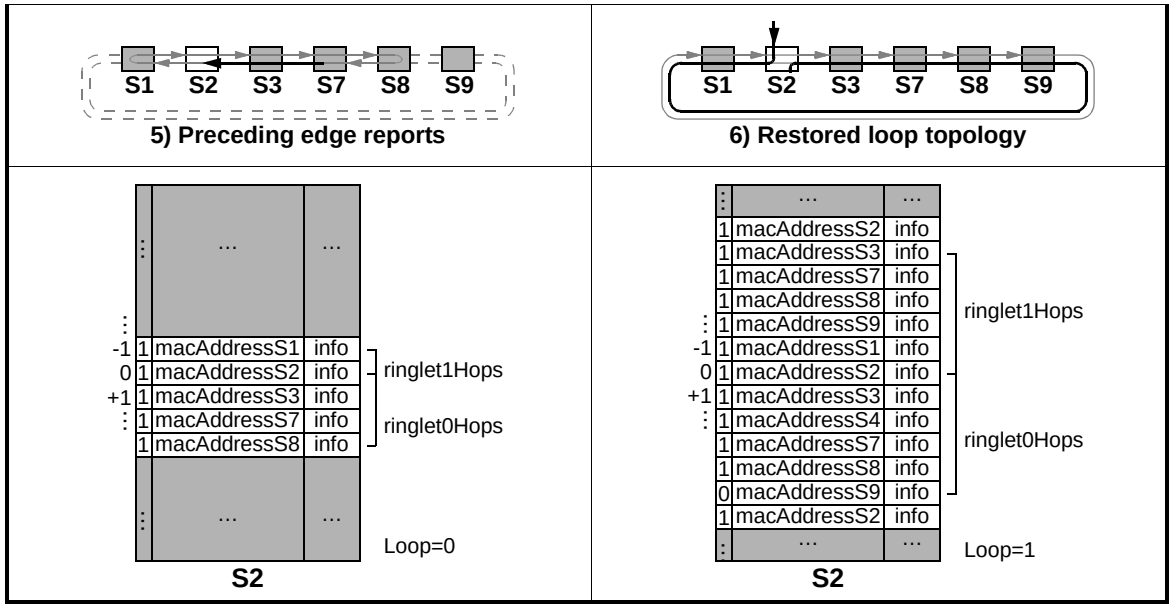
.

10.7.2.2 Topology change sequences

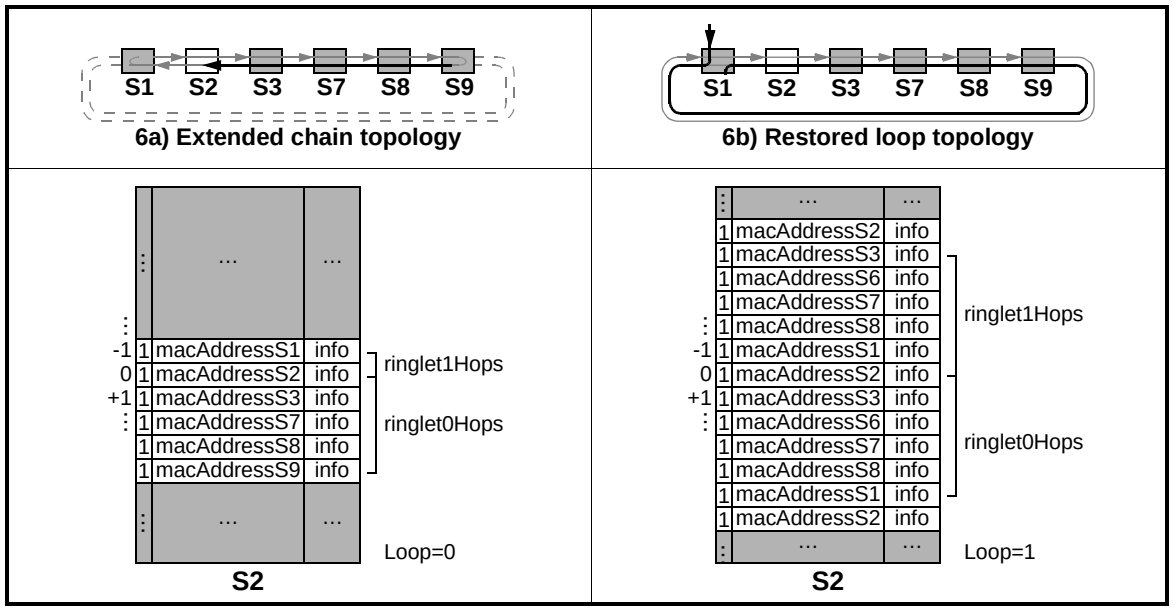
A sequence of topology changes is illustrated in Figure 10.13. A stable closed-loop topology (1) has valid positive and negative entry index values and redundant entries. A single severed link (2) restricts the database knowledge to an open-loop array, based on a pair of reports from the two edge stations. Two severed links (3) further restricts the database knowledge to a smaller open-loop array, based on a pair of reports from the two edge stations. Joining additional stations (4) requires more extensive database updates, with span extensions triggered by the remote edge-station reports.

Figure 10.13—Topology change sequences





l



The remote edge report is followed by TP frames (3) from closer stations, filling in the invalid database entries. The appearance of a connecting station (6) eliminates the possibility of edge-point reports, forcing the report of a self-generated TP frame to confirm the closed-loop topology.

Alternatively, the addition of a connecting station may happen to (6a) extend the segment before converting between chain and loop topologies. This has the advantage that the database entries are implied by the consistent incremental nature of the restoration.

10.7.3 Topology database update state machine

This subclause specifies the state machine for updating the topology database based on the following triggers:

a) Receipt of TP frames from the ring.
b) Change of local protection state information.

10.7.3.1 Inputs

rcvdTPFrame

The received TP frame, consisting of the following relevant fields:
prtw: Protection request type, west
FS, SF, SD, MS, WTR, IDLE
prte: Protection request type, east
FS, SF, SD, MS, WTR, IDLE
wscw: Wrapping status code, west
YES (1), NO (0)
wsce: Wrapping status code, east
YES (1), NO (0)
wp: Wrap preference
YES (1), NO (0)
jp: Jumbo preference
YES (1), NO (0)
ttl: Time to live
0 to 255
ri: Ringlet indicator
0, 1
sa: Source MAC address

rcvdRingletID

The ringlet ID of the ringlet on which *rcvdTPFrame* is received.

localProtInfo

The protection information of the local station, consisting of the following:
westSpanProtState: Protection state of west span
FS, SF, SD, MS, WTR, IDLE
eastSpanProtState: Protection state of east span
FS, SF, SD, MS, WTR, IDLE
westWrapStatus: Wrap status of west span (provided by protection state machine for wrapping ring only)
YES, NO
eastWrapStatus: Wrap status of east span (provided by protection state machine for wrapping ring only)
YES, NO
westNeighborMACAddress: MAC address of neighbor station on west span
eastNeighborMACAddress: MAC address of neighbor station on east span

topologyStable

Input from topology validation state machine.
TRUE, FALSE

10.7.3.2 Constants

FS
Forced switch.
SF
Signal fail.
SD
Signal degrade.
MS
Manual switch.
WTR
Wait to restore.
IDLE
Idle.

10.7.3.3 Variables

topoDB
Collection of information contained in the topology database.
localProtInfoChange
Indication that the protection information of the local station has changed.
miscablingDefectRinglet0
Indication that a miscabling defect is present on ringlet0.
miscablingDefectRinglet1
Indication that a miscabling defect is present on ringlet1.

10.7.3.4 Functions

The functions below are used in the topology database update state diagram.

IsMiscabled
The purpose of this function is to check for a miscabling defect based on a received TP frame from the neighboring station.

IsMiscabled(Frame)
((Frame.ri != *rcvdRingletID)* && (Frame.ttl == MAX_STATIONS))
Values:
TRUE: Miscabling defect present.
FALSE: Miscabling defect not present.

IndexTP
The purpose of this function is to determine the topology database index value corresponding to the received TP frame.

IndexTP(Frame)
if (*rcvdRingletID* == 1)
return (MAX_STATIONS - Frame.ttl + 1);
else
return (-1)*(MAX_STATIONS - Frame.ttl + 1);

Editors' Notes: *To be removed prior to final publication.*

The assumption below is that a station in a wrapping ring communicates its wrap status to other stations on the ring. A station in a steering ring determines its edge status based on its own topology database.

DifferenceInEntry

The purpose of this function is to check for a difference between the information contained in the received TP frame and the topology database entry at the index corresponding to that TP frame.

```
DifferenceInEntry(Frame)
i = IndexTP(Frame);
return ((topoDB[i].stationMACAddress != Frame.sa) ||
    (topoDB[i].stationWrapPref != Frame.wp) ||
    (topoDB[i].stationJumboPref != Frame.jp) ||
    (topoDB[i].westSideProtectionState != Frame.prtw) ||
    (topoDB[i].eastSideProtectionState != Frame.prte) ||
    ((ringProtectionType == WRAPPING) &&
            (topoDB[i] .westSideEdgeStatus != Frame.wscw)) ||
    ((ringProtectionType == WRAPPING) && (topoDB[i].eastSideEdgeStatus != Frame.wsce));
```

Values:

TRUE: Information in TP frame differs from corresponding topology database entry.

FALSE: Information in TP frame is the same as that in corresponding topology database entry.

DetectEdge

The purpose of this function is to determine whether an edge is present on the span connecting stations at locations *index* and (*index+1)* in the topology database. This function is for steering rings only. At least one of the two stations connected to the span must be a valid entry in the topology database.

```
DetectEdge(index)
i = index;
if ((topoDB[i].valid == 1) || (topoDB[i+1].valid == 1))
    if ((topoDB[i].eastSideProtectionState == FS) ||
            (topoDB[i].eastSideProtectionState == SF))
                return TRUE;
    else if (topoDB[i].eastSideProtectionState == SD)
            if (NoOtherSpan (i, SD))
                return TRUE;
    else if (topoDB[i].eastSideProtectionState == MS)
            if (NoOtherSpan (i, MS))
                return TRUE;
    else if (topoDB[i].eastSideProtectionState == WTR)
            if (NoOtherSpan (i, WTR))
                return TRUE;
    if ((topoDB[i+1].westSideProtectionState == FS) ||
            (topoDB[i+1].westSideProtectionState == SF))
                return TRUE;
    else if (topoDB[i+1].westSideProtectionState == SD)
```

```
                if (NoOtherSpan (i, SD))
                        return TRUE;
        else if (topoDB[i+1].westSideProtectionState == MS)
                if (NoOtherSpan (i, MS))
                        return TRUE;
        else if (topoDB[i+1].westSideProtectionState == WTR)
                if (NoOtherSpan (i, WTR))
                        return TRUE;
    return FALSE;
```

Values:
TRUE: Edge is present on span connecting stations at locations *index* and (*index+1)* in the topology database.
FALSE: Edge is not present on span connecting stations at locations *index* and (*index+1)* in the topology database.

NoOtherSpan

The purpose of this function is to determine if there is any other span in the ring (other than that connecting stations at locations *index* and (*index+1)* in the topology database) that contains the protection state of REQ or a protection state higher than REQ in the hierarchy. At least one of the stations connected to any span that is considered must be valid.

```
NoOtherSpan(index, REQ)
i = index;
for (j = (-1)*MAX_STATIONS; j = MAX_STATIONS; j++)
    if ((j != i) && ((topoDB[j].valid ==1) || (topoDB[j+1].valid == 1)))
                if ((topoDB[j].eastSideProtectionState >= REQ) ||
                        (topoDB[j+1].westSideProtectionState >= REQ))
                        return FALSE;
return TRUE;
```

Values:
TRUE: The protection request on the span connecting stations at locations *index* and (*index+1)* in the topology database is the highest protection request on the ring.
FALSE: The protection request on the span connecting stations at locations *index* and (*index+1)* in the topology database is not the highest protection request on the ring.

UpdateTopologyEntryTP

The purpose of this function is, based on the received TP frame, to update the topology database as required for a received TP frame.

```
UpdateTopologyEntryTP(Frame)
// Update topology database entry corresponding to computed index.
i = IndexTP(Frame);
topoDB[i].valid = 1;
topoDB[i].stationMACAddress = Frame.sa;
topoDB[i].stationWrapPref = Frame.wp;
topoDB[i].stationJumboPref = Frame.jp;
topoDB[i].westSideProtectionState = Frame.prtw;
topoDB[i].eastSideProtectionState = Frame.prte;
```

```
// Determine edge status based on either information in frame or on edge determination function.
if (ringProtectionType == WRAPPING)
    topoDB[i] .westSideEdgeStatus = Frame.wscw;
    topoDB[i].eastSideEdgeStatus = Frame.wsce;
else
    topoDB[i].westSideEdgeStatus = DetectEdge(i-1);
    topoDB[i].eastSideEdgeStatus = DetectEdge(i);

// Update remainder of database by clearing validity bits of entries beyond an edge, and also
// setting values for the number of hops on each ringlet.

if ((i>0) && (topoDB[i].westSideEdgeStatus == YES))
    for (j=i; j<= MAX_STATIONS; j++)
            topoDB[j].valid = 0;
    ringlet1Hops = i-1;
else if ((i>0) && (topoDB[i].eastSideEdgeStatus == YES))
    for (j=i+1; j<=MAX_STATIONS; j++)
            topoDB[j].valid = 0;
    ringlet1Hops = i;
else if ((i<0) && (topoDB[i].westSideEdgeStatus == YES))
    for (j=(-1)*MAX_STATIONS; j<i; j++)
            topoDB[j].valid = 0;
    ringlet0Hops = i;
else if ((i<0) && (topoDB[i].eastSideEdgeStatus == YES))
    for (j=(-1)*MAX_STATIONS; j<=i; j++)
            topoDB[j].valid = 0;
    ringlet0Hops = i-1;
```

UpdateTopologyEntryLocal

The purpose of this function is, based on the updated local protection information, to update the topology database.

```
UpdateTopologyEntryLocal(Info)
topoDB[0].westSideProtectionState = Info.westSpanProtState;
topoDB[0].eastSideProtectionState = Info.eastSpanProtState;
if (ringProtectionType == WRAPPING)
    topoDB[0] .westSideEdgeStatus = Info.westWrapStatus;
    topoDB[0].eastSideEdgeStatus = Info.eastWrapStatus;
else
    topoDB[0].westSideEdgeStatus = DetectEdge(-1);
    topoDB[0].eastSideEdgeStatus = DetectEdge(0);
westFormerNeighborMACAddress = Info.westNeighborMACAddress;
eastFormerNeighborMACAddress = Info.eastNeighborMACAddress;

if (topoDB[0].eastSideEdgeStatus == YES)
    for (j=1; j<= MAX_STATIONS; j++)
            topoDB[j].valid = 0;
    ringlet1Hops = 0;
if (topoDB[0].westSideEdgeStatus == YES))
    for (j=(-1)*MAX_STATIONS; j<=0; j++)
            topoDB[j].valid = 0;
```

```
        ringlet0Hops = 0;
    if ((topoDB[0].westSideProtectionState == SF) &&
        (topoDB[0].eastSideProtectionState == SF))
        ClearAllNonLocalEntries();
```

StartInstabilityDefectTimer

The purpose of this function is to start the topology instability defect timer. If the topology has not stabilized when this timer expires, a topology instability defect is triggered.

StartStabilityTimer

The purpose of this function is to start the topology stability timer. If no additional change in the topology database occurs by the expiration of this timer, the topology is considered stable and validation can proceed.

10.7.3.5 Topology database update state table

The state machine in Table 10.9 shows the procedure for updating the topology database based on the following triggers:

a) Receipt of TP frames from the ring.
b) Change of local protection state information.

Editors' Notes: *To be removed prior to final publication.*

This state machine and the underlying functions do not yet take passthrough scenarios fully into account. The updating of the database due to passthrough events could be based on marking stations invalid that are beyond the location of the detected passthrough event on the opposite ringlet. It could also be based on marking the entire topology invalid when a passthrough event is detected. In this case ringlet selection would need to choose a direction for each destination based on the closest invalid entry for that station on each direction. In addition, the impact on protection would need to be understood.

The initial state is the START (IDLE) state. In the case of any ambiguity between the text and the state machine, the state machine shall take precedence. The notation used in the state machine is described in 3.4.

When a TP frame is received from the ring, a miscabling check is performed. If this check passes, then the contents of the TP frame are checked against the corresponding entry in the topology database to determine if any of the contents of the topology database need to be modified. If not, the contents of the TP frame are discarded. If so, the topology database is modified and the appropriate timers and variables are set within the topology validation state machine.

When there is a change in local protection state information as defined from the protection state machine, the corresponding updates are also made to the topology database.

Table 10.9—Topology database update state machine

Current state		Row	Next state	
state	**condition**		**action**	**state**

START	localProtInfoChange == TRUE	1	UpdateTopologyEntryLocal();
			if (topoStability == STABLE)
			  StartInstabilityDefectTimer();
			topoStability == UNSTABLE;
			StartStabilityTimer();
			localProtInfoChange = FALSE;
				START

	rcvdTPFrame != NULL	2	--	CHECK
	--	3	--	START

CHECK	IsMiscabled(rcvdTP-Frame)	4	if (rcvdRingletID == 0)
			  miscablingDefectRinglet0 = TRUE;
			else
			  miscablingDefectRinglet1 = TRUE;
			rcvdTPFrame = NULL;
				START

	--	5	if (rcvdRingletID == 0)
			  miscablingDefectRinglet0 = FALSE;
			else
			  miscablingDefectRinglet1 = FALSE;
				UPDATE

UPDATE	!DifferenceInEn-try(rcvdTPFrame)	6	rcvdTPFrame = NULL;	START

	--	7	UpdateTopologyEntryTP();
			if (topoStability == STABLE)
			  StartInstabilityDefectTimer();
			topoStability == UNSTABLE;
			StartStabilityTimer();
			rcvdTPFrame = NULL;
				START

Row 10.9-1: If there is a change in local protection information, update the topology entry for the local station. Then start the instability defect timer (for use in the topology validation state machine for detection of the topology instability defect), set the topoStability variable to UNSTABLE, and start the stability timer (for use in the topology validation state machine for determination of when the topology has stabilized).

Row 10.9-2: If a TP frame is received from the ring, go on to the CHECK state.

Row 10.9-3: Else return to START.

Row 10.9-4: If miscabling is detected due to a ringlet identifier mismatch on a TP frame received from a neighbor station, set the appropriate miscabling defect variable to TRUE.

Row 10.9-5: Else set the appropriate miscabling variable to FALSE (enables clearing of the miscabling defect), then move on to the UPDATE state.

Row 10.9-6: If there is no difference in the relevant portions of the TP frame when compared to the appropriate entry in the topology database, return to START.

Row 10.9-7: Else update the appropriate entry in the topology database. Then start the instability timer (for use in the topology validation state machine for detection of the topology instability defect), set the topoStability variable to UNSTABLE, and start the stability timer (for use in the topology validation state machine for determination of when the topology has stabilized).

10.8 Topology validation

It can be easily determined when the topology image contained within the topology database of a station is complete and consistent by examining the image contents. Generally speaking, when the contents of the topology image show station information for each station described in the link information of another station, then the image is complete. For each ringlet, when the contents of the local topology image show that all stations on that ringlet are connected to each other in a logical ring or bus (broken ring), or show an isolated station, then the topology image is consistent.

The topology database update state machine defined in 10.7.3.5 is executed for each change in the topology image. To ensure that topology validation can be performed in a robust manner, the topology must be stable prior to the execution of most topology validation checks. The topology is stable if no changes have occurred in the topology image within the period of a topology stability timer. If the topology is unstable, then the duration required for the topology to become stable is also monitored.

Miscabling detection is done for each received TP frame from a neighbor station. The topology instability check is done based on the duration of topology instability after an initial change from a stable topology database. The other validation checks, including duplicate MAC address, neighbor inconsistency, and maximum number of stations in the ring, are done only after the topology has become stable.

A canonical form for the topology image allows all the stations to eventually arrive at the same image for the topology.

10.8.1 Miscabling detection

Each station determines which interface is associated with which ringlet and assigns the corresponding ringlet_id either through fixed mapping between hardware locations or through configuration. Each TP frame is sent separately on each ringlet, identifying the ringlet on which it is being sent. Any TP frame with a *ttl* set to MAX_STATIONS received on a ringlet different from the ringlet on which it is identified as being sent shall cause the link to be declared non-operational and trigger a miscabling detection failure. This results in alarm-triggering events being generated only from stations at which the physical misconfiguration has been detected. That shall trigger a signal fail for the link in protection. When a matched ringlet ID TP frame from its neighbor is received, that shall clear the signal fail in protection.

There is a miscabling detection defect per ringlet. This defect is triggered upon a single TP frame received with the wrong ringlet ID and *ttl*=MAX_STATIONS. In addition to the triggering of the defect, a SF condition is declared on the receive link of the span on which the frame was received. This SF condition is treated in the same way by protection as any other SF condition. If the miscabled station is the same as the last known neighbor station on this span, then WTR is applied when the miscabling defect is cleared. Otherwise it is not.

The miscabling defect clears upon the detection of a single received TP frame with the correct ringlet ID and *ttl*=MAX_STATIONS.

10.8.2 Topology instability

If the topology is determined to be unstable for more than 1 second, then a topology instability defect is triggered. This results in alarm-triggering events being generated from any station at which the topology instability has been detected.

Editors' Notes: *To be removed prior to final publication.*

The 1 second threshold for determination of topology inconsistency will be reconsidered in the context of failure declaration in standards such as ANSI T1.231. This threshold also needs to be considered in the context of passthrough events.

The value of the topology stability timer needs to be defined.

The contribution to expand the topology validation subclause will clarify the relationship between the different validation checks in 10.8.2.

The trigger to clear the topology instability defect is for there to be no change in the topology database (as defined by the topology database update state machine in 10.7.3.5) for the duration of the topology stability timer. This is also the generic criterion for declaring the topology stable.

10.8.2.1 Neighbor inconsistency

The following trigger a neighbor inconsistency alarm or indication:

Editors' Notes: *To be removed prior to final publication.*

A more detailed specification is needed for neighbor inconsistency checking. Specifically, the duration of time that a neighbor inconsistency must remain in effect after initial detection must be defined, given the wide possible range of TLV frame periods.

a) East station's west neighbor MAC address is not equal to local station MAC address, or
b) West station's east neighbor MAC address is not equal to local station MAC address.

The zero MAC address is an exceptional case for this inconsistency check. The inconsistency check shall be performed when the neighbor information is updated due to a new received TLV frame. An alarm is triggered to the operator if this neighbor inconsistency is detected.

10.8.2.2 Duplicate MAC address

Editors' Notes: *To be removed prior to final publication.*

Upon clearing of a duplicate MAC address defect at a station, does the station automatically begin sourcing traffic on the ring again, or is this operator commanded?

A MAC address is the unique identification for a station on an RPR ring. If a station receives a TP frame from another station with the same MAC address as its MAC address, a defect is triggered to the operator for this duplicate MAC address. In addition to the triggering of an alarm, a station that detects a duplicate of its own MAC address shall cease transmitting frames either by going into passthrough, or by taking itself administratively down, the choice of which is an implementation decision.

The determination that a duplicate MAC address scenario actually exists is nontrivial, particularly in passthrough scenarios. For this reason, the duplicate MAC address check takes place only after the topology is determined to be stable. The intent of the duplicate MAC address check at a station A is to detect if there is a station duplicating the MAC address A. There will not be any checking of the presence of other duplicate MAC address pairs at station A.

The conditions that trigger a duplicate MAC address defect are:

1) MAC address i hops away on ringlet0 != MAC address (num stations on ring - i) hops away on ringlet1 (loop through all valid stations in topology; for this check to pass there cannot be “gaps” in the topology where there are invalid stations.)
2) Neighbor MAC address on either ringlet == station's own source MAC

The above conditions consider only stations marked as valid in the topology.

The above checks handle the following scenarios:

LOOP:

a) A0=B=C=D=A1=E=A0 (check 1)
b) A0=B=C=D=A1=E=F=G=A0 (check 1)
c) A0=B=C=D=A1=B=C=D=A0 (undetectable)
d) A0=A1=A2=A0 (check 2)

Condition c) above is undetectable using check 1 (or check 2) because the two halves of the ring are mirror images of each other.

CHAIN:

a) E=A0=B=C=D=A1 (check 1)
b) A0=B=C=D=A1=E=F=G=A2 (check 1)
c) A0=A1=A2 (check 2)

The duplicate MAC address defect clears when the topology again becomes stable and when, for that topology, the duplicate MAC address check passes.

10.8.2.3 The number of stations exceed MAX_STATIONS

If the total number of entries in the topology database exceeds MAX_STATIONS, a defect shall be triggered to the operator. This defect can be triggered under two conditions. In a ring or loop configuration, the number of stations from which TP frames are received can be detected to equal up to (MAX_STATIONS + 1) through receipt of frames from a station 255 hops away. In a chain configuration, the number of stations from which TP frames are received can be detected to equal up to (2*MAX_STATIONS + 1) through receipt of frames from stations 255 hops away on each direction.

Again, this check is only applied when a "stable" topology has been reached.

The maximum number of stations exceeded defect clears when the topology again becomes stable and when, for that topology, the number of valid stations does not exceed MAX_STATIONS.

10.8.3 Defects specified

The following defects are specified in this clause:

a) Ringlet ID mismatch in 10.8.1: critical severity,
b) Topology instability in 10.8.2: critical severity,
c) Neighbor inconsistency in 10.8.2.1: critical severity,
d) Duplicate MAC address, local station, in 10.8.2.2: critical severity,
e) Maximum number of stations exceeded, in 10.8.2.3: critical severity.

10.8.4 Topology validation state machine

This subclause specifies the state machine for validating the topology database. At startup this state machine most likely begins with topology in an unstable state. If the topology stabilizes, the topology moves into the stable state. If all validation checks then pass, the topology moves into the valid state. If any validation check fails, the topology moves into the invalid state and then directly into the unstable state so that the validation process can be repeated.

Editors' Notes: *To be removed prior to final publication.*

The details of the topology validation state machine need to be written out in terms of inputs, variables, and function definitions for the next version of the draft. These details will be based on the framework that has been included in this version of the clause.

Table 10.10—Topology validation state machine

Current state		Row	Next state	
state	**condition**		**action**	**state**
START	topoStability == UNSTABLE	1	--	UNSTABLE
	--	2	--	STABLE
UNSTA-BLE	Instability defect timer expires	3	topoInstabilityDefect = TRUE; StartInstabilityDefectTimer();	UNSTABLE
	Stability timer expires	4	topoInstabilityDefect = FALSE; ClearInstabilityDefectTimer();	STABLE
STABLE	topoStability == UNSTABLE	5	--	UNSTABLE
	Neighbor consistency check fails	6	neighborConsistencyDefect = TRUE;	INVALID
	Neighbor consistency check passes	7	neighborConsistencyDefect = FALSE;	
	Max number of stations check fails	8	maxStationsDefect = TRUE;	INVALID
	Max number of stations check passes	9	maxStationsDefect = FALSE;	
	Duplicate MAC check fails	10	duplicateMACDefect = TRUE;	INVALID
	--	11	duplicateMACDefect = FALSE; Set some topology database values (such as reachability, ringTopologyType, ringProtec-tionType, etc.), throw out invalid entries if a loop is detected, etc.	VALID
INVALID	topoStability == UNSTABLE	12	--	UNSTABLE
VALID	topoStability == UNSTABLE	13	--	UNSTABLE

Row 10.10-1: Upon startup, if the topology is unstable proceed directly to the UNSTABLE state.

Row 10.10-2: Upon startup, if the topology is stable proceed directly to the STABLE state.

Row 10.10-3: If the instability defect timer expires, declare a topology instability defect. Stay in the UNSTABLE state and start the instability defect timer again.

Row 10.10-4: If the topology stability timer expires, the topology is stable. Clear the topology instability defect and move to the STABLE state.

Row 10.10-5: If there is a change in the topology database, move from the STABLE state to the UNSTABLE state.

Row 10.10-6: If the neighbor consistency check fails, declare a neighbor inconsistency defect. Move to the INVALID state.

Row 10.10-7: If the neighbor consistency check passes, clear the neighbor inconsistency defect. Continue on to perform more checks.

Row 10.10-8: If the maximum number of stations check fails, declare a maximum number of stations exceeded defect. Move to the INVALID state.

Row 10.10-9: If the maximum number of stations check passes, clear the maximum number of stations exceeded defect. Continue on to perform more checks.

Row 10.10-10: If the duplicate MAC address check fails, declare a duplicate MAC address defect. Move to the INVALID state.

Row 10.10-11: If the duplicate MAC address check passes, clear the duplicate MAC address defect. Set appropriate global values within the topology database that are set only based on a valid topology. Move to the VALID state.

Row 10.10-12: If there is a change in the topology database, move from the INVALID state to the UNSTABLE state..

Row 10.10-13: If there is a change in the topology database, move from the VALID state to the UNSTABLE state..

10.9 TP frame handling

Upon triggers defined in 10.9.1, a station broadcasts a TP frame to all stations on both ringlets. The TP frame contains information about the local station relevant to connectivity, including its links to its neighbors. When a station receives a TP frame, it updates its local topology image.

The TP frame contains the ringlet ID on which it is sent. When a station receives a TP frame with a *ttl* set to MAX_STATIONS (indicating the frame has traveled exactly one hop), it verifies that the ringlets of both stations are connected with the same ringlet ID value (as indicated in the received frame and by local configuration).

The TP frame, like all RPR frames, contains the source MAC address of the station from which it is sent as part of the RPR header. When a station receives a TP frame with a *ttl* set to MAX_STATIONS (indicating the frame has traveled exactly one hop), it verifies that it knows who its neighbor is.

Upon triggers defined in 10.11.1, a station broadcasts a TLV frame to all stations on both ringlets. When a station receives a TLV frame, it also updates its local topology image.

10.9.1 Transmit state machine

This subclause specifies the state machine for determining when TP frames are sent on the ring from a station, and related requirements.

The TP frame is broadcast:

a) On the initial start of RPR topology discovery.
b) At any point that a station determines that there is a change in protection information, as defined in the protection state machine defined in 10.6.3. A change in protection information refers to a change in the reported protection status on a receive link monitored by the station, or a change in the wrap status of the span of which that link is a part. A change in the wrap preferred bit shall not trigger transmission of the TP frame; this information is disseminated through the regular periodic transmissions of the TP frame.
c) At any point that a station detects a new station on the ring.
d) Periodically.

Upon each of the above triggers, one copy of the TP frame is sent on each ringlet.

The TP frames are sent on a bi-level periodic timer, starting with the fast period. The fast TP frame period is configurable from 1 ms to 20 ms with 1 ms resolution and a default value of 10 ms. The fast rate timer is used for the first INIT_FAST_MSG_COUNT frames sent after triggering, including the initial copy of the message that is sent. INIT_FAST_MSG_COUNT equals 8. The slow TP frame period is configurable from 50 ms to 10 seconds with 50 ms resolution and a default value of 100 ms. Each time a TP frame is triggered by a change in the information contained in the frame or upon initialization, the fast TP frame period is used, followed by the slow TP frame period until a new frame is triggered from the station.

10.9.1.1 Inputs

transmitTPFrame
The trigger for immediately transmitting TP frames and restarting transmission using the fast protection timer. The trigger is set to true as described in 10.9.1, and as per the protection state machine defined in Table 10.6. The fields in the payload of the TP frame are set based on the values set in the topology database defined in 10.4.4.
Values:
TRUE: The fast protection timer has expired.
FALSE: The fast protection timer has not expired.

Editors' Notes: *To be removed prior to final publication.*

The setting of transmitTPFrame also needs to be included in the appropriate topology state machines.

10.9.1.2 Constants

INIT_FAST_MSG_COUNT
The maximum number of frames to be transmitted on the fast protection timer, defined in 10.9.1.

10.9.1.3 Variables

fastTPFrameCount
The number of TP frames remaining to be transmitted on the fast protection timer.

10.9.1.4 Functions

IsFastTPTimeExpired
To indicate if the fast protection timer has expired.
Values:
TRUE: The fast protection timer has expired.
FALSE: The fast protection timer has not expired.

IsSlowTPTimeExpired
To indicate if the slow protection timer has expired.
Values:
TRUE: The slow protection timer has expired.
FALSE: The slow protection timer has not expired.

SendTPFrame
Inserts a copy of the TP frame into both MAC control queues (one per ringlet). The MAC control queue is defined in Clause 6.

SetTPFastTimer
Sets the protection fast timer to *fastTimerValue*. The range is defined in 10.9.1.

SetTPSlowTimer
Sets the protection slow timer to *slowTimerValue*. The range is defined in 10.9.1.

10.9.1.5 TP frame transmit state table

The TP frame transmit state machine specified in Table 10.11 implements the functions necessary for determining when TP frames are sent on the ring from a station. The initial state is the START state. Optional rows are shaded in grey. In the case of any ambiguity between the text and the state machine, the state machine shall take precedence. The notation used in the state machine is described in 3.4.

Table 10.11—TP frame transmit state machine

Current state		Row	Next state	
state	**condition**		**action**	**state**

START	transmitTPFrame == TRUE	1	SendTPFrame();
			fastTPFrameCount = INIT_FAST_MSG_COUNT - 1;
			transmitTPFrame = FALSE;
			SetTPFastTimer();
				WAITFAST

	--	2	SendTPFrame(); SetTPSlowTimer();	WAITSLOW
WAITFAST	IsFastTPTimeExpired()	3	SendTPFrame(); fastTPFrameCount = fastTPFrameCount -1;	CHECK
	transmitTP-Frame==TRUE	4	--	START
	--	5	--	WAITFAST
CHECK	fastTPFrameCount == 0	6	SetTPSlowTimer();	WAITSLOW
	fastTPFrameCount != 0	7	SetTPFastTimer();	WAITFAST
WAIT-SLOW	IsSlowTPTimeExpired()	8	SendTPFrame();	WAITSLOW
	transmitTP-Frame==TRUE	9	--	START
	--	10	--	WAITSLOW

Row 10.11-1: Use of the fast timer for sending INIT_FAST_MSG_COUNT TP frames is based on transmitTPFrame being set to TRUE by the protection or topology database update state machines (due to a change in topology or protection information). Use of the fast timer is set up, then transmitTPFrame is set to FALSE.

Row 10.11-2: Even if transmitTPFrame is FALSE, use of the slow timer for sending TP frames is set up.

Row 10.11-3: When the fast protection timer expires, a TP frame is sent on each ringlet and the number of TP frames remaining to be sent on the fast timer is decremented.

Row 10.11-4: Change in topology or protection information causes return to START for re-start of TP frame transmissions on the fast timer.

Row 10.11-5: Remain waiting in WAITFAST state.

Row 10.11-6: INIT_FAST_MSG_COUNT TP frames have been sent (on each ringlet) on the fast timer, so now the slow timer will be used for subsequent transmissions.

Row 10.11-7: INIT_FAST_MSG_COUNT TP frames have not yet been sent (on each ringlet) on the fast timer, so the fast timer will be used for the next transmission.

Row 10.11-8: When the slow protection timer expires, a TP frame is sent on each ringlet.

Row 10.11-9: Change in topology or protection information causes return to START for re-start of TP frame transmissions on the fast timer.

Row 10.11-10: Remain waiting in WAITSLOW state.

10.9.2 Receive

The receipt of this frame on either ringlet from any station causes the MAC control sublayer to update its current local topology image, as long as the sequence number check has passed. The sequence number check is described in the flow chart shown in Figure 10.14.

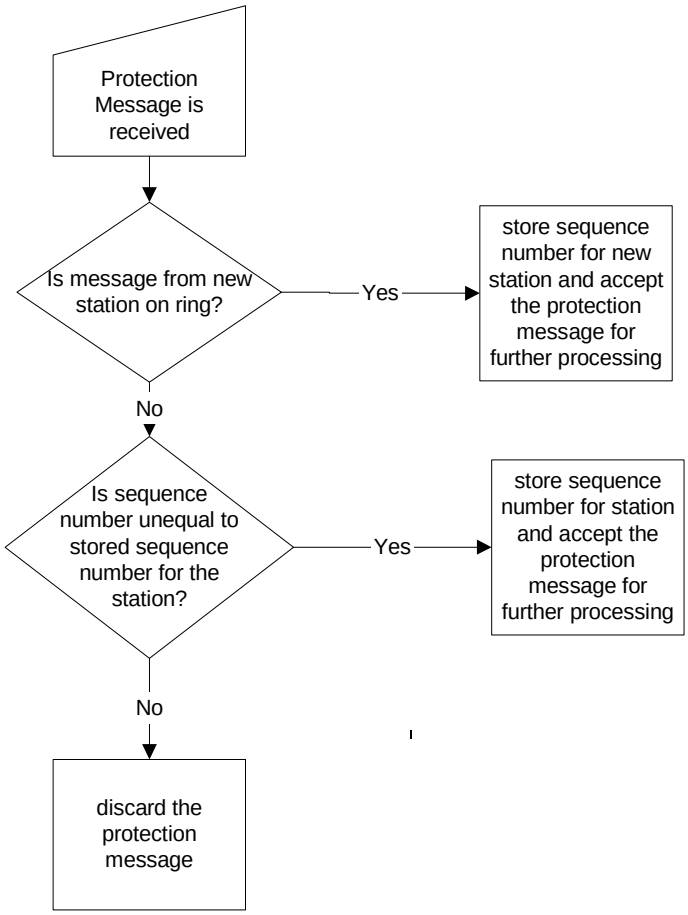


Figure 10.14—Flow chart for sequence number check

The sequence number is used to enable stations to determine that the contents of a TP frame received from a given source station has changed. If the sequence number on the received TP frame is different from the sequence number stored for the source station, then the TP frame is accepted for further processing.

The sequence number stored for a station is cleared when the entry for the station in the topology database is marked invalid or cleared from the database.

> ***Editors' Notes:*** *To be removed prior to final publication.*
>
> *Flow chart will be modified to FrameMaker format in an upcoming version of the draft.*

The maximum value of the 6-bit sequence number field is MAX_SEQNUM, which equals 63.

In addition, the following contents of the TP frame are made available to the protection state machine running on each side of the station:

a) *sa* (source address)
b) *ttl* (time to live)
c) *ri* (ringlet identifier)
d) *wscw* (wrap status code, west)
e) *wsce* (wrap status code, east)
f) *prtw* (protection request type, west)
g) *prte* (protection request type, east)
h) *wp* (wrap preferences)

The protection state machine differentiates received TP frames into two types. One type is those received on the "short" path from the neighbor station directly connected to the interface on which the protection state machine is running. These frames are used, for example, in unwrapping scenarios when a span is coming back into operation. The criteria for a frame to be considered a short path frame from the neighbor station are shown in Figure 10.15.

All other TP frames are used to determine if protection conditions such as manual switch or wait to restore are preempted by protection conditions reported from elsewhere on the ring.

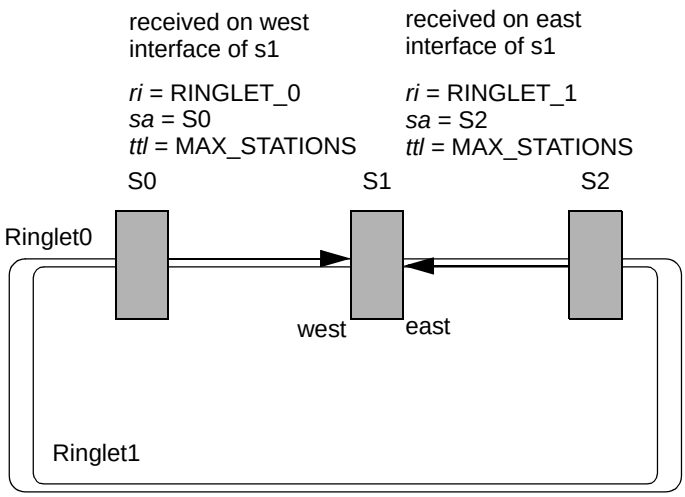


Figure 10.15—Conditions for a frame to be "short" path and from the neighbor

The receipt of this frame on the same ringlet as which it was sent (as indicated in the ringletID field) from a neighbor station (as determined by *ttl*==MAX_STATIONS) causes the MAC control sublayer to validate and (if needed) update the identity of its neighbor, as long as the sequence number check has passed.

The receipt of this frame on a ringlet other than that on which is was sent (as indicated in the ringletID field) from a neighbor station (as determined by *ttl*==MAX_STATIONS) causes the MAC control sublayer to discard the frame, place the link upon which the frame was received into a non-operational state, and generate a miscabling error.

10.9.3 Handling of TP frames during protection

A station in wrapped protection state shall not wrap a TP frame, and shall strip it after receiving it. The *we* (wrap eligible) bit in the RPR header shall be set to zero for TP frames to ensure that they are not wrapped, as described in 8.2.2.5. TP frames shall continue to be delivered and received on links that are in non-idle protection states.

10.10 typeLengthValue (TLV) protocol

10.10.1 TLV contents

Within the topology database, the information per station entry that is filled in by information contained in TLVs (not including vendor-specific information) is shown in Table 10.12.

The values of each entry are:

a) the MAC address of the station described by the entry, *stationMACAddress*,
b) station name, *stationName*,
c) the MAC address of the neighboring station connected to the west span of the station, *westNeighborMACAddress*,

d) the MAC address of the neighboring station connected to the east span of the station, *eastNeighborMACAddress*,
e) the station weight on ringlet0, *ringlet0Weight*,
f) the station weight on ringlet1, *ringlet1Weight*,
g) the reserved subclassA0 bandwidth on ringlet0, *ringlet0ResBW*,
h) the reserved subclassA0 bandwidth on ringlet1, *ringlet1ResBW*.

Table 10.12—TLV contents in topology and status database

local MAC	station name	west neighbor's MAC	east neighbor's MAC	ringlet0 station weight	ringlet1 station weight	ringlet0 reserved sub-classA 0 band-width	ringlet1 reserved sub-classA 0 band-width
00-10-A4-97-A8-DE	Los Angeles	00-10-A4-97-A8-EF	00-10-A4-97-A8-BD	1	5	50	40
00-10-A4-97-A8-EF	Denver	00-10-A4-97-A8-AC	00-10-A4-97-A8-DE	3	3	100	80
00-10-A4-97-A8-AC	Seattle	00-10-A4-97-A8-BD	00-10-A4-97-A8-EF	5	1	150	120
00-10-A4-97-A8-BD	Portland	00-10-A4-97-A8-DE	00-10-A4-97-A8-AC	1	5	200	160

Some rules related to updates of the topology database resulting from TLV frames are:

a) The topology database does not age stored information from TLV frames.
b) When a TLV frame of a given type is received, all stored information contained in this type of TLV frame is replaced.
c) All TLV information received from station forms part of the topology database entry for that station.

10.10.2 TLV frame format

The TLV frame is a variable length frame. The TLV frame is not used for discovery of the physical topology of the ring, but rather is used to convey additional information that needs to be reported by a station to the rest of the ring on a slower timeframe, such as reserved bandwidth configuration information. The TLV frame is used for reporting changes in station status that are not as time critical as those reported in the TP frame.

The TLV frame format is outlined in Figure 10.16.

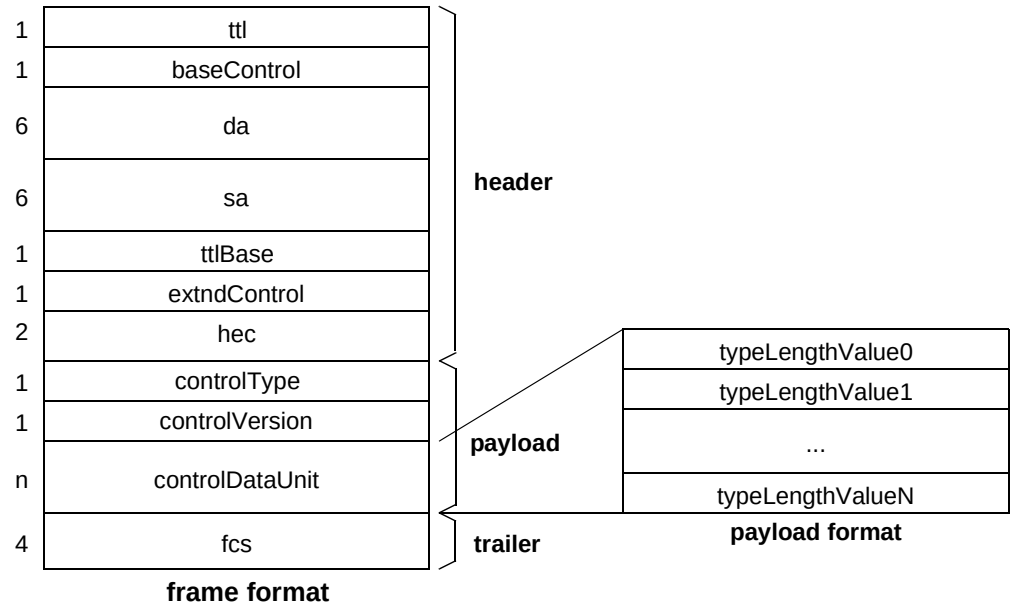


Figure 10.16—topology frame format

The header portion of Figure 10.16 corresponds to the RPR header of the RPR control frame defined in 8.3. The *controlVersion* and *controlType* fields correspond to the fields with those names that are part of the RPR payload of the RPR control frame. The fields specific to the TLV frame comprise the *controlDataUnit* portion of the payload, as described in the following subclauses.

There are two types of TLV frames: a station TLV frame and a vendor-specific TLV frame. As shown in Table 10.13, all TLV types other than the vendor-specific TLV shall be usable within the station TLV frame. As shown in Table 10.14, only the vendor-specific TLV shall be usable within the vendor-specific TLV frame.

The station TLV frame is sent as a RPR Control frame with a *controlType* value of CT_STATION_TLV in Table 8.7. The vendor-specific TLV frame is sent as a RPR Control frame with a *controlType* value of CT_VENDOR_TLV in Table 8.7. Both types of TLV frames are sent as a broadcast frame on all ringlets, with a *ttl* (time to live) of MAX_STATIONS to remove topology dependencies, are stripped from the ring by the source station, and have a *sa* (source address) set to the actual MAC of the sending station.

10.10.3 TLV entries

A typeLengthValue (TLV) encoding scheme is used to encode additional station information in station TLV and vendor-specific TLV frames. The TLVs may appear in the topology frame in any order.

The general format of a TLV is shown in Figure 10.17.

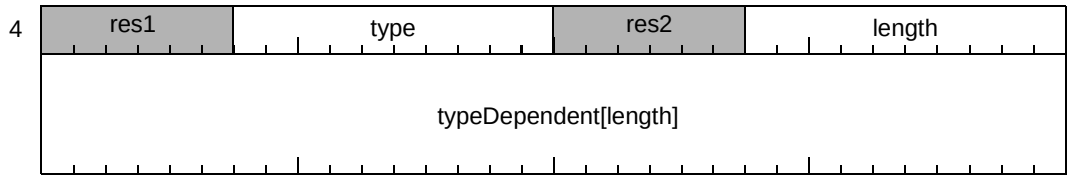


Figure 10.17—typeLengthValue format

The 6-bit *res1* field shall be reserved. The 10-bit *type* field specifies the TLV type, as specified in Table 10.13. The 6-bit *res2* field shall be reserved. The 10-bit *length* field specifies the length of the following *typeDependent* value information (shown in Table 10.13 and in Table 10.14), in bytes. The length of any *typeDependent* information shall be less than 1024 bytes.

Table 10.13—*type* encoding for station TLV frame

Value	Name	Size	sub clause	Description
0	—	—	—	Reserved
1	TLV_WEIGHT	2	10.10.4.1	Weights of stations on each ringlet
2	TLV_STATION_BW	4	10.10.4.2	Total reserved classA0 bandwidth (per ringlet)
3	TLV_NEIGHBOR	12	10.10.4.3	Neighbor address
4	TLV_STATION_NAME	255	10.10.4.4	Station name (an ASCII string)
5-1023	—	—	—	Reserved

NOTES
1—N is the number of other stations (one less than the total stations)
2—S is the number of characters in the character string

Table 10.14—*type* encoding for vendor-specific TLV frame

Value	Name	Size	sub clause	Description
0-5	—	—	—	Reserved
6	TLV_VENDOR_SPEC	8+D	10.10.4.5	Nonstandard values (possibly vendor dependent)
8-1023	—	—	—	Reserved

NOTE
1—D is the number of dependent data bytes

10.10.4 Defined TLV encodings

The following subclauses define the TLV encodings supported by this standard. Any other *type* values are reserved.

10.10.4.1 Weight TLV

The weight TLV encodes the station weight in each ringlet. If a station has a weight not equal to 1 on either ringlet, it shall advertise the weight. If a weight TLV is not included in the station TLV messages sent by a given source station, then it is assumed that the station's weight on each ringlet is equal to 1.

The weight TLV is identified by its distinctive *type*=TLV_WEIGHT value (see 10.10.3); its payload consists of two classC weight parameters, as illustrated in Figure 10.18.

The source of the information contained in the weight TLV is the station weight per ringlet described in 9.3.8. These parameters are configurable per station, and are reported to all other stations on the RPR ring using the weight TLV. The consumer of the information contained in the weight TLV is the fairness module, that needs this information to compute the sum of the weights of active stations in 9.1.17.3.

10.10.4.1.1 type

The *type* field is set to TLV_WEIGHT.

10.10.4.1.2 length

The *length* field is set to the size corresponding to TLV_WEIGHT in Table 10.13.

10.10.4.1.3 Payload format

The format for the payload of the Weight TLV is shown in Figure 10.18.

Figure 10.18—Weight TLV payload format

The 8-bit *ringlet0Weight* and 8-bit *ringlet1Weight* fields specify the weight of the station on ringlet0 and ringlet1 respectively.

10.10.4.2 Station bandwidth TLV

The station bandwidth TLV encodes the reserved subclassA0 bandwidth on the transmit direction on each ringlet used by the source station, and is mandatory. The station bandwidth TLV is identified by its distinctive *type*=TLV_STATION_BW value (see 10.10.3); its payload consists of two subclassA0 bandwidth parameters, as illustrated in Figure 10.19.

The source of the information contained in the station bandwidth TLV is the subclassA0 bandwidth reserved by the sending station for its transmissions on each ringlet. This reservation is made through configuration. These parameters are reported to all other stations on the RPR ring using this TLV. Each station then sums

up the reserved bandwidth reported by each station to obtain the total reserved bandwidth on each ringlet. The consumer of the total reserved bandwidth on each ringlet is the fairness module, that needs this information to determine the total reclaimable bandwidth in 9.3.15.

10.10.4.2.1 type

The *type* field is set to TLV_STATION_BW.

10.10.4.2.2 length

The *length* field is set to the size corresponding to TLV_STATION_BW in Table 10.13.

10.10.4.2.3 Payload format

The format for the payload of the station bandwidth TLV is shown in Figure 10.19..

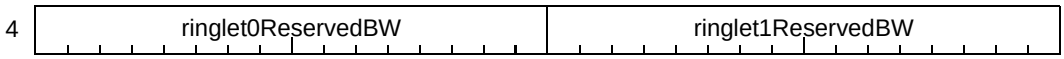


Figure 10.19—Station bandwidth TLV payload format

The 16-bit *ringlet0ReservedBW* and 16-bit *ringlet1ReservedBW* fields specify the reserved subclassA0 bandwidth of the station's transmissions on ringlet0 and on ringlet1, respectively.

The reserved bandwidth shall use the same normalization as the *controlValue* field in fairness frames defined in 9.3.8, with WEIGHT equal to 1. The purpose of this normalization is so that reserved bandwidth is made available to the fairness module in the most convenient form possible.

10.10.4.2.4 Computation of total reserved bandwidth

The total reserved bandwidth on a given ringlet (ringlet0 in the below code example) is computed as follows:

```
void computeTotalResBW()
{
    totalRinglet0ReservedBW = ringlet0ResBW[0];

    for (hopCount = 1; hopCount < NUM_STATIONS; hopCount++)
        totalRinglet0ReservedBW +=
            stationEntry[hopCount].ringlet0ResBW[NUM_STATIONS - hopCount];

}
```

In the above example:

- *totalRinglet0ReservedBW* is the total reserved bandwidth on ringlet0,
- *ringlet0ResBW[0]* is the reserved bandwidth of this station for its transmissions on ringlet0,
- *hopCount* is the number of hops away on ringlet0 that the station referred to by *stationEntry* is from this station,
- *NUM_STATIONS* is the total number of stations on the ring as determined from the topology database,

— *ringlet0ResBW[NUM_STATIONS - hopCount]* is the reserved bandwidth of the station *hopCount* hops away on ringlet0 that needs to be added to *totalRinglet0ReservedBW.*

In the event that the total reserved bandwidth exceeds *LINK_RATE* on either ringlet, a reserved shaping failure condition is declared. This results in an alarm-triggering event being generated from any station that detects the failure condition.

10.10.4.3 Neighbor address TLV

The neighbor address TLV encodes the MAC addresses of the neighbors of the sending station, and is mandatory. The neighbor address TLV is identified by its distinctive *type*=TLV_NEIGHBOR value (see 10.10.3); its payload consists of two 48-bit fields, as illustrated in Figure 10.20.

10.10.4.3.1 type

The *type* field is set to TLV_NEIGHBOR.

10.10.4.3.2 length

The *length* field is set to the size corresponding to TLV_NEIGHBOR in Table 10.13.

10.10.4.3.3 Payload format

The format for the payload of the neighbor address TLV is shown in Figure 10.20.

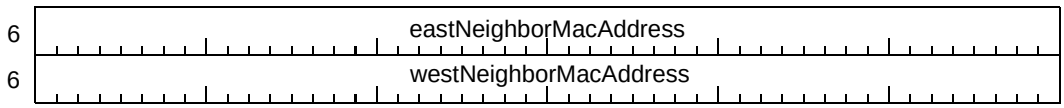


Figure 10.20—Neighbor address TLV payload format

The 48-bit *eastNeighborMACAddress* field carries the MAC address of the neighbor connected to the east interface of the station. If the east neighbor's MAC address is unknown, *eastNeighborMACAddress* shall be zero.

The 48-bit *westNeighborMACAddress* field carries the MAC address of the neighbor connected to the west interface of the station. If the west neighbor's MAC address is unknown, *westNeighborMACAddress* shall be zero.

10.10.4.4 Station name TLV

The station name TLV encodes the operator assigned station name *stationName*, and is optional. The station name follows the definition of the RFC 1907 sysName object, a zero to 255 character arbitrary printable ASCII string. The station name TLV is identified by its distinctive *type*=TLV_STATION_NAME value (see 10.10.3); its payload is illustrated in Figure 10.21.

The source of the information contained in the station name TLV is the configurable station name described in 10.7. This station name is reported to all other stations on the RPR ring using the station name TLV. The consumer of the information contained in the station name TLV is the topology module, that needs this information to fill in the station name field in the topology and status database shown in Table 10.7.

10.10.4.4.1 type

The *type* field is set to TLV_STATION_NAME.

10.10.4.4.2 length

The *length* field is set to the size corresponding to TLV_STATION_NAME in Table 10.13.

10.10.4.4.3 Payload format

The format for the payload of the station name TLV is shown in Figure 10.21.

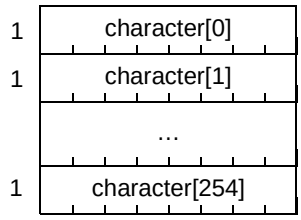


Figure 10.21—Station name TLV payload format

The *character[0]* through *character[254]* fields identify the first through last ASCII characters corresponding to the station's name (a null-character termination is not provided). The acceptable subset of ASCII characters supported is the set defined in IEEE 1212, with the exception that only characters of space and above are supported.

10.10.4.5 Vendor specific TLV

The Vendor specific TLV encodes the vendor specific information of the station, and is optional. This TLV is identified by its distinctive *controlType*=CT_VENDOR_TLV and its distinctive *type*=TLV_VENDOR_SPEC value (see 10.10.3); its payload is illustrated in Figure 10.22.

The source of the information contained in the vendor specific TLV is a collection of vendor specific parameters. This information is reported to all other stations on the RPR ring using the vendor specific TLV. The consumer of the information contained in the vendor specific TLV depends on the parameters and is outside the scope of this standard.

10.10.4.5.1 type

The *type* field is set to TLV_VENDOR_SPEC.

10.10.4.5.2 length

The *length* field is set to the size corresponding to TLV_VENDOR_SPEC in Table 10.13.

10.10.4.5.3 Payload format

The format for the payload of the vendor specific TLV is shown in Figure 10.22.

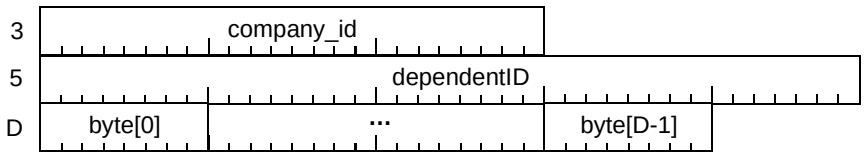


Figure 10.22—Vendor specific TLV payload format

The first 64 bits specify a globally-unique EUI-64 identifier, consisting of *company_id* and *dependentID* components. The 24-bit *company_id* field is supplied by the IEEE/RAC for the purposes of identifying the organization supplying the (unique within the organization) 40-bit *dependentID*.

The format and function of the remaining *D* bytes are dependent on the initial 64-bit identifier. The exact definition of the function of this vendor specific data is outside of the scope of this standard.

10.11 TLV frame handling

10.11.1 When generated

The station TLV and vendor-specific TLV frames shall be broadcast:

a) On the initial start of RPR topology discovery.
b) Periodically. If the vendor-specific TLV frame is used, it shall be sent at the same frame period as the station TLV frame.

The TLV frames are sent on a periodic timer configured to *tlvTimerValue*. The TLV frame period is configurable from 50 ms to 10 seconds with 50 ms resolution and a default value of 1 second.

Editors' Notes: *To be removed prior to final publication.*

A technically binding comment required that the word "exactly" be removed before 50 ms resolution in the line above, because it conveys no meaning. A statement of 50 ms implies 50 ms.

10.11.1.1 Effect of receipt

The receipt of this frame on the same ringlet as which it was sent (as indicated in the ringletID field) from any station causes the MAC control sublayer to update its current local topology image.

A station receiving a TLV frame with one or more TLVs of unknown type shall ignore those TLVs and process the rest of the TLV message as if those TLVs were not present.

10.11.1.2 Handling of TLV frames during protection

A station in wrapped protection state shall not wrap a TLV frame, and shall strip it after receiving it. The wrap eligible (WE) bit in the RPR header shall be set to zero for TLV frames to ensure that they are not

wrapped, as described in 8.2.2.5. TLV frames shall continue to be delivered and received on links that are in non-idle protection states.

10.12 Protocol Implementation Conformance Statement (PICS) proforma for Clause 10[1]

Editors' Notes: *To be removed prior to final publication.*

This PICS proforma is at present incomplete. No valid items should be assumed to be present in this proforma. Conformance of any type to any portion of this clause shall not be claimed on the basis of any item in this section.

This standards draft shall not be considered to be complete until this PICS proforma is complete.

The editors estimate that the level of completeness of this PICS proforma is 0%.

10.12.1 Introduction

The supplier of a protocol implementation that is claimed to conform to Clause 10, Topology discovery and protection, shall complete the following Protocol Implementation Conformation Statement (PICS) proforma.

A detailed description of the symbols used in the PICS proforma, along with instructions for completing the same, can be found in Annex A of IEEE Std 802.1Q-1998.

10.12.2 Identification

10.12.2.1 Implementation identification

Supplier[a]	
Contact point for enquiries about the PICS[a]	
Implementation Name(s) and Version(s)[a,c]	
Other information necessary for full identification—e.g., name(s) and version(s) for machines and/or operating systems; System Name(s))[b]	

a—Required for all implementations

b—May be completed as appropriate in meeting the requirements for the identification.

c—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).

[1]*Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

10.12.2.2 Protocol summary

Identification of protocol standard	IEEE Std 802.17-200x, Resilient Packet Ring Access Method and Physical Layer Specifications, Topology discovery and protection
Identification of amendments and corrigenda to this PICS proforma that have been completed as part of this PICS	
Have any Exception items been required? No [ ] Yes [ ] (The answer Yes means that the implementation does not conform to IEEE Std 802.17-200x.)	

Date of Statement	

10.12.3 Major capabilities/options

Item	Feature	Subclause	Value/Comment	Status	Support
<cap>	<Required capability>	<xref>	<capability> supported.	M	Yes [ ]
<opt>	<Optional capability>	<xref>	<option> supported.	O	Yes [ ] No [ ]
MD	Management Data Interface	12.1.1.3	Management data interface supported.	O	Yes [ ] No [ ]

10.12.4 PICS tables for Clause 10