Annex F

(normative)

802.1D and 802.1Q Bridging conformance

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

The following should be updated in the terms and definitions:

bidirectioned flooding -- a frame forwarding transfer involving sending two flooding frames, one on each run, where the each frame is directed to distinct adjacent stations.

bidirectional flooding -- a frame forwarding transfer involving sending two flooding frames, one on each run, where the each frames is directed to the same station.

unidirectioned flooding -- a frame forwarding transfer involving sending a flooding frame in the downstream direction, and that frame is directed to the station's upstream neighbor.

unidirectional flooding -- a frame forwarding transfer involving sending a flooding frame in the downstream direction, and that frame is directed to its sending station.

remote unicast -- A transfer directed to a specific RPR station with the intent of forwarding a stripped version of that frame to a distinct remote node.

local unicast -- A transfer directed to a specific RPR end station.

multidrop -- A multicast, broadcast, or flooded frame.

F.1 Flooding techniques

F.1.1 TTL aging

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

The following discussion on TTL aging better belongs in the overview (with details in clause 6), but is being retained in this annex until new placement location is confirmed.

The RPR protocols include the use of a *timeToLive* (TTL) field within the frame header. The primary function of the *timeToLive* field is to limit frame lifetimes to no more than 256 hops, whereupon an error is logged and the frame is discarded. A secondary function is to allow other stations to determine the hop-count distance from the source of the frame.

This *timeToLive* field is set to 255 when the frame is first transmitted and is decremented when passing through stations. Depending on the wrapping condition of the ring, the TTL is decremented on one or both runs, as illustrated in the left and right sides of Figure F.1 respectively.

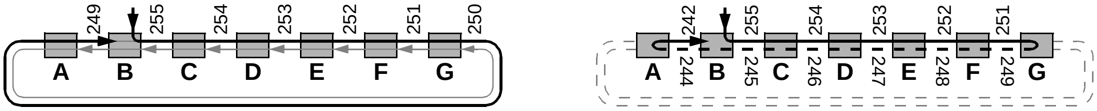


Figure F.1—TTL aging

At the wrap point, the *timeToLive* field is decremented by two, due to the apparent presence of an attachment on both runs. Implementations that elect to pass wrapped traffic through only one of the two attachment are responsible for decrementing *timeToLive* by two (rather than one) when passing through that wrapped attachment.

The uniform processing of the TTL allows stations to easily determine the number of hops between the frame source and destination. For fairness frames, the hop-count determines the choke-point location; for discovery frames, hop-count is the index for storing information into the topology-database array.

F.1.2 Multicast/broadcast forwarding

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

The following discussion on multicast/broadcast forwarding probably belongs in the overview (with details in clause 6), but is being retained in this annex until new placement location is confirmed:

The most basic multicast/broadcast distribution techniques involves circulating a frame through all stations on the ring. The forwarding techniques for multicast/broadcast transfers are the same as those described for flooded frames, in the left sides of Figure F.8 through Figure F.19. However, there is no learning or remote-unicast dependencies associated with multicast/broadcast transfers, so the illustrations on the right portions of Figure F.8 through Figure F.19 do not apply.

NOTE—Stations are not expected to optimize the efficiency of multicast forwarding. To reduce complexities, they are expected either support unidirectional multicasts or to forward multicasts and flooded frames in the same fashion.

F.1.3 Flooding bridge transfers

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

The following flooding text probably belongs in the overview, since its mostly informative.

Transparent bridging requires a form of multidrop called flooding. Multidrop protocols (flooding, multicast, and broadcast) require the inclusion of additional information, beyond that included within the client-visible Ethernet frame. That information includes local destination station identifier (DSID) and local source station identifier (SSID) along with other maintenance/control fields. These addresses assist in scoping the range of the multidrop distribution and suppressing undesirable duplicates that might otherwise be generated.

Bridges use the DSID/SSID addresses to flood (a flood is a type of broadcast) remote frames for delivery to all all bridge clients, as illustrated in the left side of Figure F.2. Flooding involves multidrop transmissions between a single source station and all other stations. Flooding a ring involves the unidirectional or bidirectional transmission of frames, normally when the destination of the frame is unknown or for MAC address learning purposes.

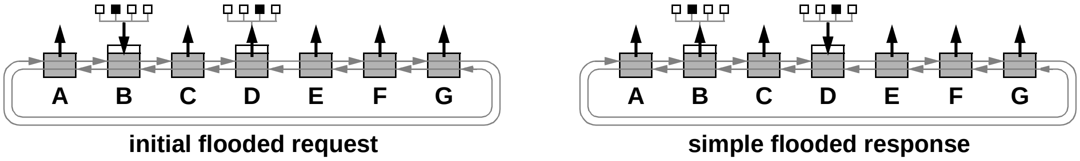


Figure F.2—Basic bridge flooding

Basic-bridging stations maintain simplicity by handling all requests and returned responses the same, as illustrated in the right side of Figure F.2. This also avoids overheads associated with maintaining and utilizing RPR forwarding tables.

Enhanced-bridging stations improve efficiencies by maintaining and utilizing RPR forwarding tables. Enhanced stations are able to *learn* from previous transfers, as illustrated in the left side of Figure F.3. The learn improves the efficiency of the returned response frame, due to its unicast (not flooded) and shortest-path nature. From that response, the local source can learn its source address, so that following frames can also utilize a remote unicast request (as opposed to flooding again).

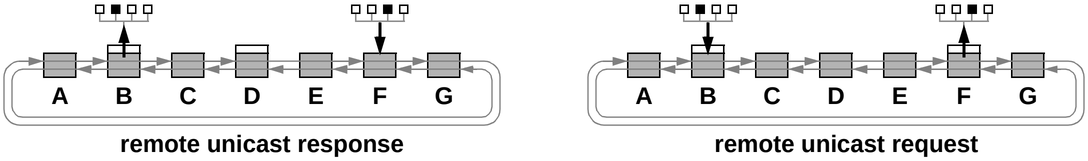


Figure F.3—Enhanced bridge unicasts

Ordering constraints mandate that flooded and related remote-unicast transfers flow over the same path. The term related refers to fames with an identical set of {sourceMacAddress/destinationMacAddress,VLAN} identifiers. Flowing over the same path is necessary to maintain ordering, without invoking an inefficient flush between floods and related remote-unicast transfers.

For unidirectioned/unidirectional flooding, the potential performance impact of this constraint can be severe, in that the worst case path-length nearly doubles over that associated with bidirectional flooding. To avoid that potential performance impact, enhanced bridges are expected to support bidirectional or bidirectioned flooding.

F.1.4 Remote frame formats

Learned remote transfers involve prepending of ringlet-local identifiers (*DSID* and *SSID*) as well as remote endpoint identifiers (*destinationMacAddress* and *sourceMacAddress*). Figure F.4 illustrates how these remote addresses are prepended with RPR routing information, to ensure their reliably RPR delivery.

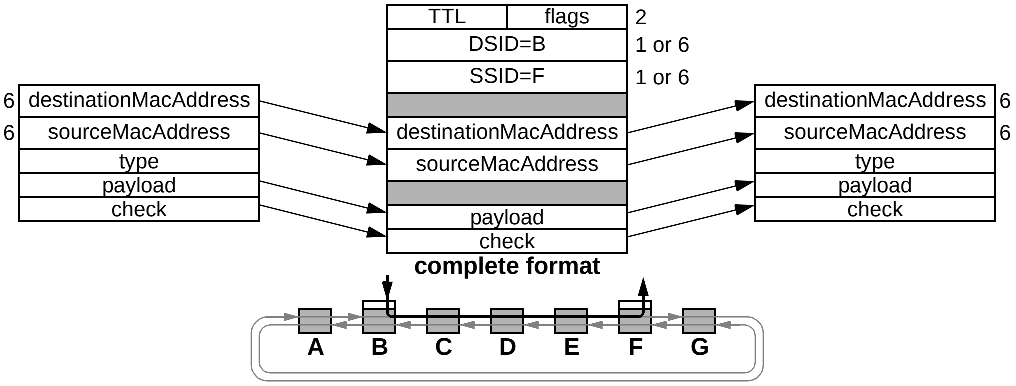


Figure F.4—Remote frame formats

.

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

Depending on the selected design option (F.7, F.6, F.5, or F.4): the *DSID* and *SSID* values may be 1-byte or 6-byte fields; the *destinationMacAddress* and the *sourceMacAddress* are always 6-byte fields, but may be located in the header or payload.

F.1.5 Multidrop flow notation

A variety of remote-transfer flows are illustrated in following subclauses, as illustrated in Figure F.5. Downward and upward arrows identify client-to-MAC and MAC-to-client transfers respectively. Downward end-of-flow curves identify locations where frames are stripped. The 'x' marker at the end of an error identifies locations where frames are discarded, due to detected inconsistency errors.

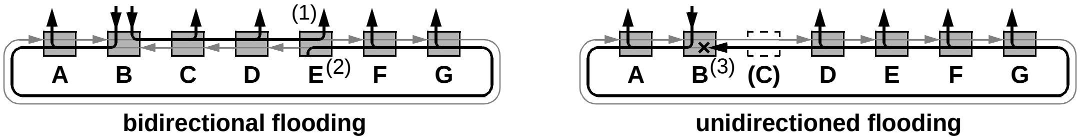


Figure F.5—Multidrop receive operations

The frame header contains sufficient information to invoke (1) FLOOD_COPY or (2) FLOOD_TOSS operations, as illustrated on the top and bottom runs within stationE on the left side of Figure F.5. A frame deletion operation (3) may also occur, as illustrated within stationB on the right side of Figure F.5. The deletion operations is invoked by consistency-check errors, rather than decoded frame-header values.

F.1.6 Flooding alternatives

Several flooding alternatives are provided, as illustrated in Table F.1. Within this table, the first-through-last entries are the most-through-least complex, but also the least-through-most efficient. Although stations are expected to use the second (bidirectional) alternative, they may use any of the four listed alternatives. Within this table, the *Reference* column provides a cross-reference to the applicable descriptive subclause.

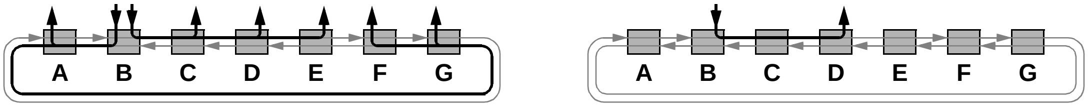
Table F.1—Flooding alternatives

Target0	Target1	Reference	Name	Description
midNear0	midNear1	F.1.7	bidirectioned	Two floods, to distinct midpoint stations
minPoint	-na-	F.1.8	bidirectional	Two floods, but to the same midpoint station
upstream	-na-	F.1.9	unidirectioned	One downstream flood to the upstream station
self	-na-	F.1.10	unidirectional	One flood to the source (*DSID*=*SSID*)

F.1.7 Bidirectioned flooding

F.1.7.1 Bidirectioned unwrapped flooding

Bidirectioned flooding of a ring involves concurrent transmissions on both runs, typically directed to a pair of mid-point station, as illustrated in the left side of Figure F.6. After the destination station has been learned, a more efficient remote-unicast transfer may be used, as illustrated in the right side of Figure F.6.



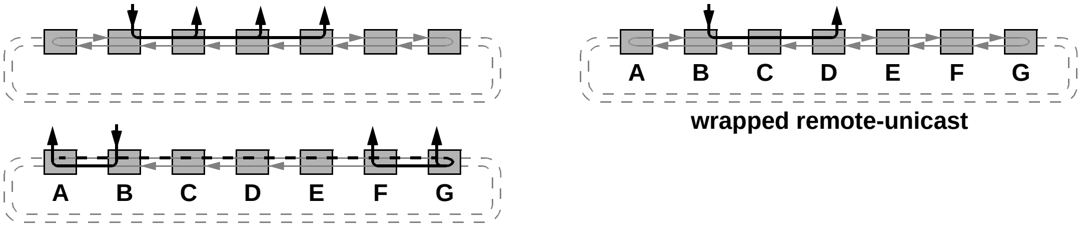
ring flooding learned remote-unicast

Figure F.6—Bidirectioned ring flooding

A FLOOD_TOSS receive operation is specified for leftside as well as rightside transfers.

F.1.7.2 Bidirectioned wrapped flooding

Bidirectioned wrapped flooding involves concurrent transmissions to the common midpoint station, as illustrated in the left side of Figure F.7. After the destination station has been learned, a more efficient remote-unicast transfer may be used, as illustrated in the right side of Figure F.7.



wrapped remote-unicast

wrapped flooding

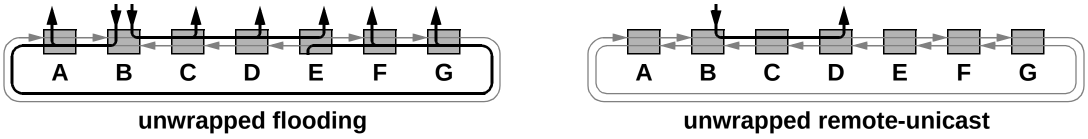
Figure F.7—Bidirectioned wrapped-chain flooding

Again, a FLOOD_TOSS receive operation is specified for leftside as well as rightside transfers.

F.1.8 Bidirectional flooding

F.1.8.1 Bidirectional unwrapped flooding

Bidirectional flooding of a ring involves concurrent transmissions on both runs, typically directed to a midpoint station, as illustrated in the left side of Figure F.8. After the destination station has been learned, a more efficient remote-unicast transfer may be used, as illustrated in the right side of Figure F.8.



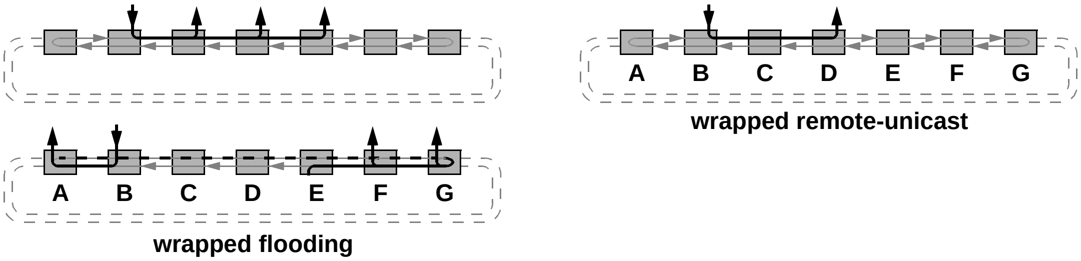
unwrapped flooding unwrapped remote-unicast

Figure F.8—Bidirectional unwrapped flooding

A FLOOD_COPY receive operation is specified for one of the bidirectional transfers; a FLOOD_TOSS operation is specified for the other. In this example, copy-strip and strip operations are specified for right-side and left-side transfers respectively.

F.1.8.2 Bidirectional wrapped flooding

Bidirectional wrapped flooding involves concurrent transmissions to the common midpoint station, as illustrated in the left side of Figure F.9. After the destination station has been learned, a more efficient remote-unicast transfer may be used, as illustrated in the right side of Figure F.9.



wrapped remote-unicast

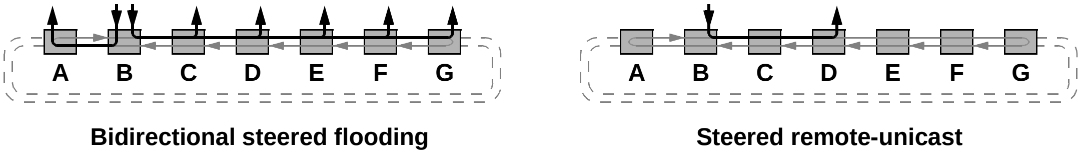
wrapped flooding

Figure F.9—Bidirectional wrapped flooding

A operations are not changed by the wrapping condition. In this example, FLOOD_COPY and FLOOD_TOSS operations are specified for right-side and left-side transfers respectively.

F.1.8.3 Bidirectional steered flooding

Bidirectional steered flooding involves concurrent transmissions with distinctive nonadjacent stationA and stationG wrap-point destinations, as illustrated in the left side of Figure F.10. After the destination station has been learned, a more efficient remote-unicast transfer may be used, as illustrated in the right side of Figure F.10.



Bidirectional steered flooding Steered remote-unicast

Figure F.10—Bidirectional steered-chain flooding

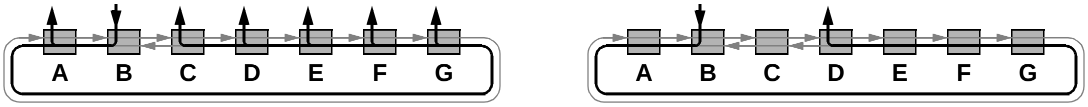
A FLOOD_COPY receive operation is specified for both of the bidirectional steered transfers.

F.1.9 Unidirectioned flooding

F.1.9.1 Unidirectioned unwrapped flooding

Flooding complexity can be reduced by flooding in one of two possible directions, rather than both. Simple load-balancing techniques could involve hashing the *sourceMacAddress*, *VLAN*, and *priority* fields to generate the *select* value that determines the left-path and right-path directions. Consistent hashing of flood and directed frames is assumed.

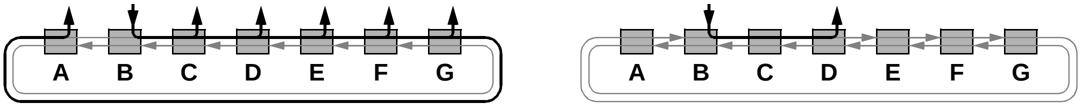
Unidirectined leftside ring flooding involves a single leftside transmission directed to the rightside station. Related unlearned-flooded and learned-remote-unicast transfers are sent over the same path, to maintain ordering constraints, as illustrated in Figure F.11. A FLOOD_COPY receive operation is specified.



unidirectioned leftside flooding **unidirectioned leftside remote-unicast**

Figure F.11—Unidirectioned leftside ring transfers

Unidirectioned rightside ring flooding involves a single rightside transmission directed to the leftside station. Related flooded and remote-unicast transfers are sent over the same path, to maintain ordering constraints, as illustrated in Figure F.12. A FLOOD_COPY receive operation is specified.

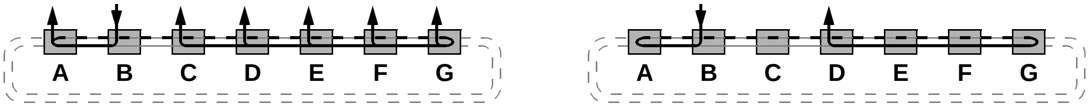


unidirectioned rightside flooding **unidirectioned rightside remote-unicast**

Figure F.12—Unidirectioned rightside ring transfers

F.1.9.2 Unidirectioned wrapped flooding

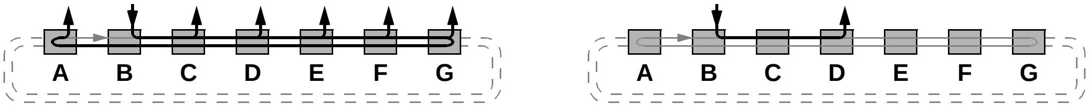
Unidirectioned leftside wrapped flooding relies on the wrap capability at the endpoints, as illustrated in Figure F.13. A FLOOD_COPY receive operation is specified.



leftside flooding **leftside remote-unicast**

Figure F.13—Unidirectioned leftside wrapped flooding

Undirectioned rightside wrapped flooding relies on the wrap capability at the endpoints, as illustrated in Figure F.14. A FLOOD_COPY receive operation is specified.



rightside flooding **rightside remote-unicast**

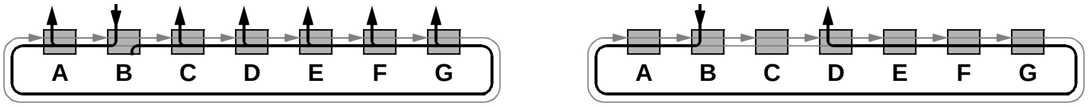
Figure F.14—Unidirectioned rightside wrapped flooding

F.1.10 Unidirectional flooding

F.1.10.1 Unidirectional unwrapped flooding

For unidirectioned flooding, complexity is further reduced by not needing to know destination topology information, as opposed to nearest-neighbor or shortest-path information associated with unidirectional and bidirectional flooding.

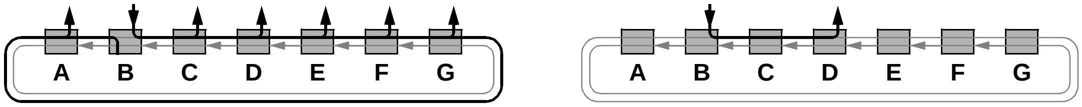
Unidirectional rightside ring flooding involves a single leftside transmission directed to the source station. Related unlearned-flooded and learned-remote-unicast transfers are sent over the same path, to maintain ordering constraints, as illustrated in Figure F.15. A FLOOD_TOSS receive operation is specified.



leftside flooding **leftside remote-unicast**

Figure F.15—Unidirectional leftside ring transfers

Unidirectional rightside ring flooding involves a single rightside transmission directed to the leftside station. Related unlearned-flooded and learned-remote-unicast transfers are sent over the same path, to maintain ordering constraints, as illustrated in Figure F.16. A FLOOD_TOSS receive operation is specified.

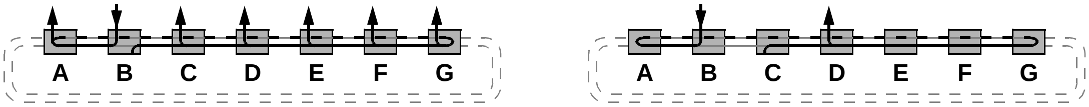


rightside flooding **rightside remote-unicast**

Figure F.16—Unidirectional rightside ring transfers

F.1.10.2 Unidirectional wrapped flooding

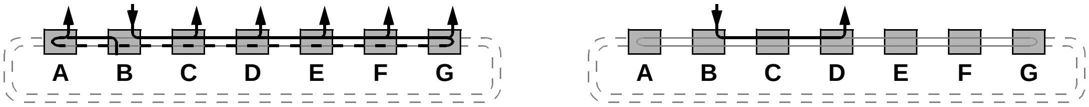
Unidirectional leftside wrapped flooding relies on the wrap capability at the endpoints, as illustrated in Figure F.17. A FLOOD_TOSS receive operation is specified.



leftside flooding **leftside remote-unicast**

Figure F.17—Unidirectional leftside wrapped flooding

Unidirectional rightside wrapped flooding relies on the wrap capability at the endpoints, as illustrated in Figure F.18. A FLOOD_TOSS receive operation is specified.



rightside flooding **rightside remote-unicast**

Figure F.18—Unidirectional rightside wrapped flooding

F.1.11 Flooding constraints

Based on its flooding constraints, clause 6 of IEEE Std 802.1D-1998 architecture is summarized as follows:

a) Loss: A frame may be lost due to many reasons including topology change (subclause 6.3.2).
b) Misordering. Frame misordering is not allowed under any conditions (subclause 6.3.3).
c) Duplication. Frame duplication is not allowed under any conditions (subclause 6.3.4).

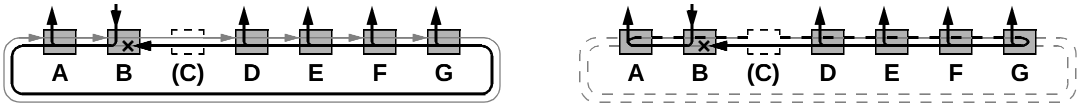
The misordering constraint can be met by having the client force a flush of in-progress transimissions before topology changes are allowed to affect the forwarding parameters (see F.3). In normal operation, the duplication constraint is met by flooding in nonoverlapping directions, as described in F.1.8. In abnormal operation, such as during topology changes the duplication constraint mandate consistency checks based on destination-station identifier (DSID) and source-station identifiers (SSID) values, as described in F.1.8.

F.1.12 Duplicate deletion stations

Duplicate deletion techniques are described in the following subclauses.

F.1.12.1 Source deletion

Source deletion is effective for unidirectional flooding, providing protection against a destination station loss shortly before or during the flood transmission. Source deletion involves stripping the flood operation based on an unexpected match between the station's MAC address and the frame's SSID address, illustrated by the x marks in Figure F.19.



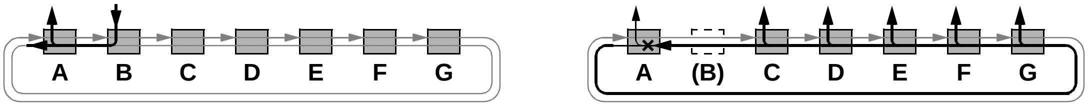
unidirectional leftside ring flooding **unidirectional leftside wrapped flooding**

Figure F.19—Source deletions

Rightside deletion is performed in a similar (not illustrated) fashion.

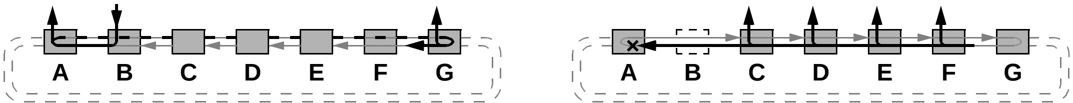
F.1.12.2 Timeout deletion

Timeout deletion is effective for unidirectioned flooding, providing protection against a source station loss during the flood transmission. Timeout deletion involves stripping the flood operation based on an unexpected difference between actual and observed *timeToLive* values, illustrated by the *x* marks in Figure F.20 and Figure F.21. Rightside deletion (not illustrated) is performed in a similar fashion.



ring flooding starts **ring flooding completes**

Figure F.20—Ring timeout deletions



flooding starts **flooding completes**

Figure F.21—Wrapped timeout deletions

Timeout deletions involves discarding possibly duplicate frames when all of the following conditions are met:

a) *frame.runID==attachment.runID*
b) *(256-frame.timeToLive)>database.distanceFromSSID*
c) *(256-frame.timeToLive)!=(database.distanceFromSource+database.stationsOnRing)*

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

These rules would also appear to work if the magnitude comparison of (b) were replaced by a not-equal comparison. The benefits and deficits of such an approach are under active investigation.

F.1.12.3 Interval deletion

Duplicate deletions involves enabling multiple destination strippers, to take the place of a single absent destination-station stripper, as illustrated in Figure F.22. Within the figure, the flood-deletion capable stations are colored white.

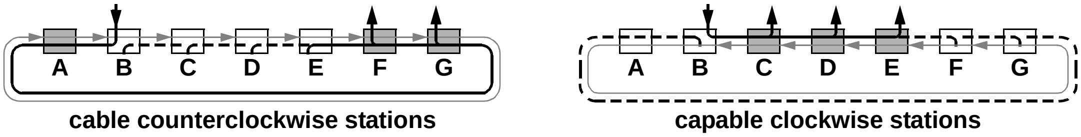


Figure F.22—Interval deletion stations

For counterwise floods, stationD strips station-B-to-stationE floods when stationE disappears; stationC strips station-B-to-stationE floods when stationE and stationD disappear; stationB strips station-B-to-stationE floods when stationE, stationD, and stationB disappear.

For clockwise floods, stationF strips station-B-to-stationE floods when stationE disappears; stationG strips station-B-to-stationE floods when stationE and stationF disappear; stationA strips station-B-to-stationE floods when stationE, stationF, and stationG disappear; stationB strips station-B-to-stationE floods when stationA, stationE, stationF, and stationG disappear.

Interval-based duplicate deletion involves discarding frames under any of the conditions listed below. The effect is that duplication is avoided despite the disappearance of nearly half of the stations.

a) DSID invalid. The DSID is not within the topology database.
b) SSID invalid. The SSID is not within the topology database.
c) Interval invalid. The following conditions are all true.
    1) *frame.runID==attachment.runID*.
    2) *database.distanceFromSSID>database.distanceFromDSID*.

Interval deletions are necessary to support bidirectioned or bidirectional flooding, so that duplicates can be deleted when the destination station disappears, as illustrated in Figure F.23.

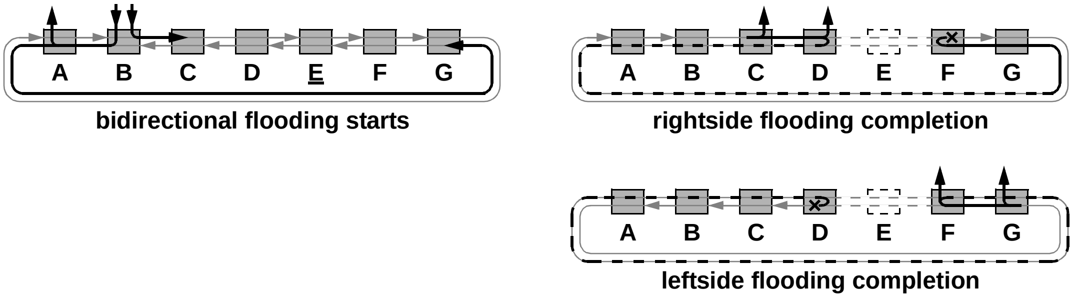

A
B
C
D
E
F
G
bidirectional flooding starts
A
B
C
D
E
F
G
rightside flooding completion
A
B
C
D
E
F
G
leftside flooding completion

Figure F.23—Bidirectional interval deletions

F.1.13 Duplicate deletion rules

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

The following text currently represents a supplement to clause #6; it should be reconciled and placed therein.

The duplicate-deletion involves frame processing as follows:

a) A frame is deleted when the *runID* equals from the *runID* of the attachment, and any of the following deletion conditions apply:
    1) Source deletion (see F.1.12.1).
    2) Timeout deletion (see F.1.12.2).
    3) Interval deletion (see F.1.12.3).
b) If not deleted, the copying of multidrop frames depends on their type:
    1) The FLOOD_COPY multidrop frames are copied to the client.
    2) The FLOOD_TOSS multidrop frame copied to clients on all but DSID-matching stations.
c) The multidrop frames with a matching DSID address are then stripped.

F.2 Flushing residual frames

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

Flushing operations are expected to be invoked by the client in response to protection or topology-change information provided by the MAC.

The client-to-MAC interface is required to support such state-change information, but the details have not specified and should probably be specified within other clauses (not this one).

With a ring topology, flushing involves sending flushing frames in left-path and right-path directions, as illustrated in Figure F.24. Although illustrated as two separate transmissions for the sake of clarity, these transmissions are expected occur concurrently.

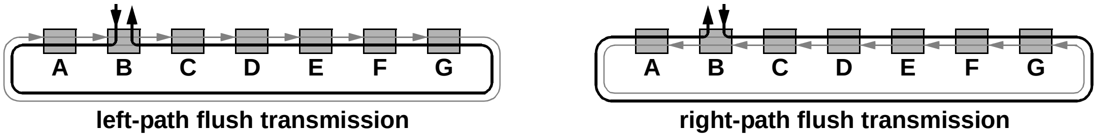


Figure F.24—Ring flush operations

The flushing frames are marked as wrappable, so that the client-visible portion of the flushing completes in the same fashion, whether applied to a ring (illustrated in Figure F.25) or wrapped topology illustrated in Figure F.25. As before, these transmissions are expected occur concurrently.

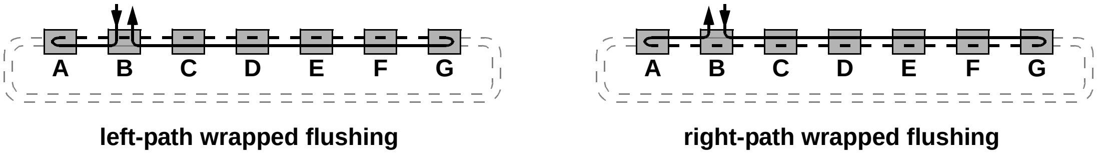


Figure F.25—Wrapped flushing operation

F.3 Dynamic protection and topology changes

F.3.1 Severed link effects

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

Several well-versed individuals have pointed out that no flush is required before transitioning from wrapped to unwrapped states. The reader should therefore be wary that this text is subject to change, to incorporate these observations, when editing time becomes available.

F.3.1.1 Link failures

A link failure effects the routing (1) of packets that would normally pass over the affected link. A station may elect to wrap quickly, returning packets (2a) on the opposing run rather than discarding them at the failed link, as illustrated in the left of Figure F.26.

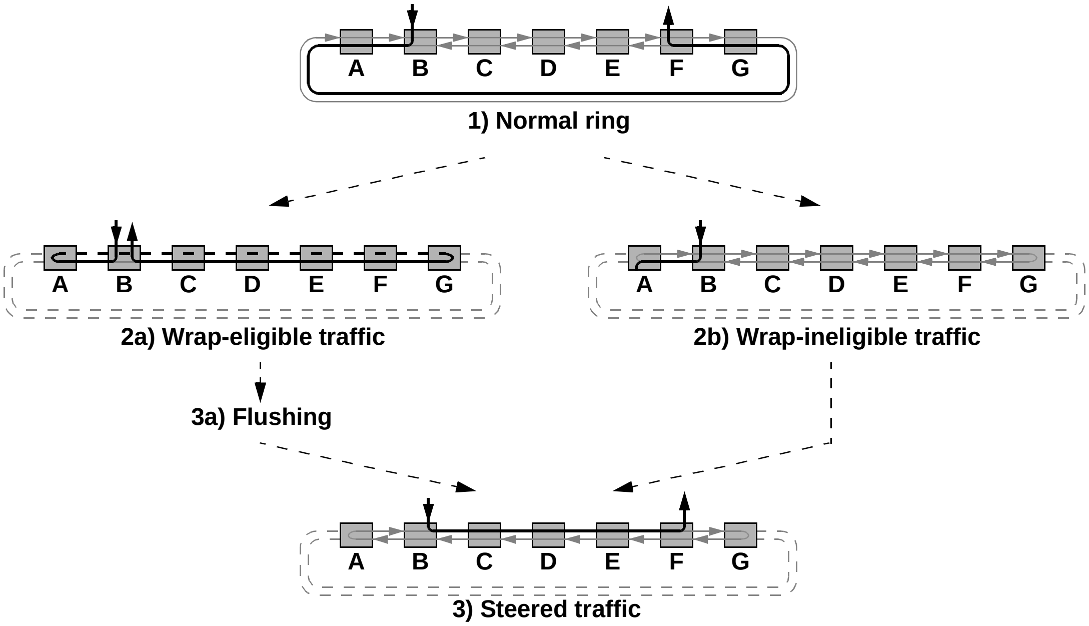


Figure F.26—Protection

Wrapping minimizes packet loss during the protection event, but consumes excessive link bandwidth until steering is enabled thereafter. The time delay required to perform flushing (3a, see F.2) also limits the usefulness of wrapping for latency-sensitive class-A traffic, since time-sensitive traffic cannot be sent during the flush operation.

Wrapping is expected to be a transient state, as steering is more efficient and part of the failed-link recover protocols (which are ultimately invoked when the failed link operation is restored). To maintain packet ordering when switching between wrapping and steering modes, outstanding traffic must be flushed before the change occurs. Packet flushing involves sending a non-class-A packet to one's self, along the wrapping path. All packets are known to be delivered, and switching between wrapping and steering is therefore safe, when this packets returns to its source.

If only steering is employed, some traffic will continue to be lost (2b) until intermediate stations become aware of the ring failure, and begin transmitting traffic on both ringlets. However, because the data packets are discarded at the failed link, no flushing is required before steering (4) is invoked.

F.3.1.2 Link recovery

A severed link (1) is expected to be recovered, whereupon dual ringlet connectivity (2) becomes available. Efficient utilization of these ringlets involves flushing outstanding traffic (3) before redirecting traffic (4) in the normal unwrapped-ring fashion, as illustrated in Figure F.27.

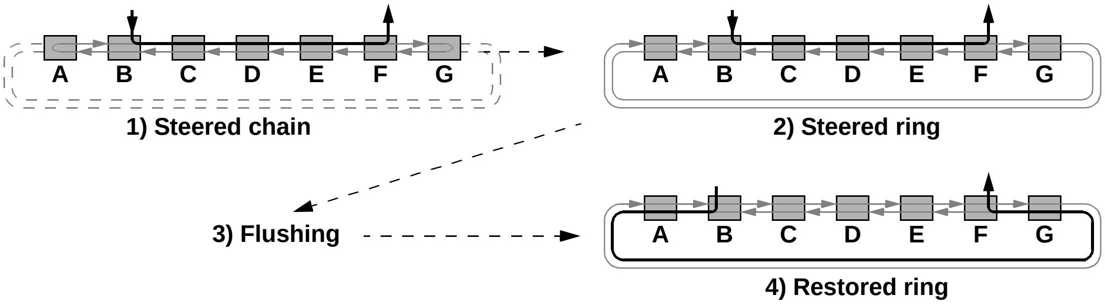


Figure F.27—Protection steering

Link recovery from a wrapped mode is not supported. Instead, wrapped rings are converted to steered rings (see F.3.1.1), whereupon the aforementioned link-recovery techniques can be used.

F.3.2 Detaching and attaching stations

On a normal ring (1), one or more of the attached stations (2) can be detached, so that fewer stations appear connected to the ring, as illustrated in Figure F.28. A flushing step is required (2a) before revising shortest-path forwarding tables for the new topology (3).

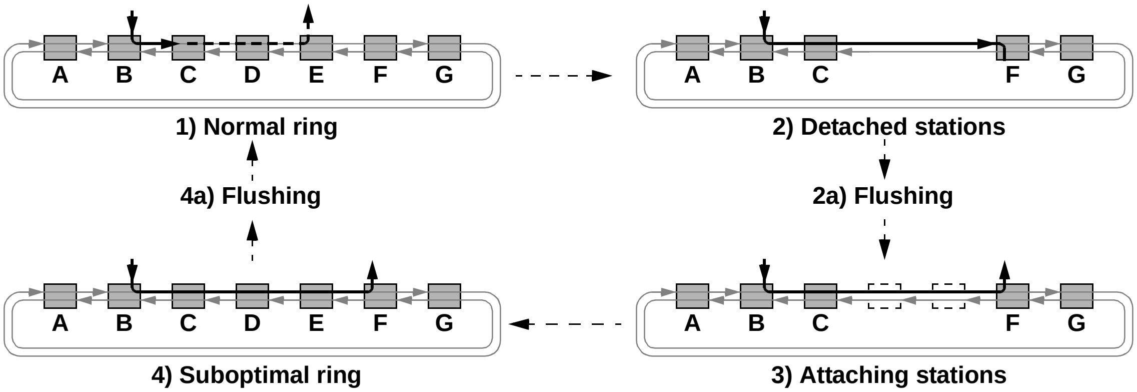


Figure F.28—Detaching and attaching stations

If intermediate stations (4) are then attached, the optimality of run-selection protocols is compromized. A flushing step is required (4a) before revising shortest-path forwarding tables for the new topology (1).

F.3.3 Split and joined chains

A second failure (1) in a ring (on a single failure in a chain) can form two distinct rings, as illustrated in Figure F.29. After separation, steered frames sent towards the severed chain (2) are discarded at the wrap point. A flush is necessary to rid the interconnect of the disjoint stationE/stationF/stationG traffic, to prevent missordering if the stationA-to-stationG list is nearly simultaneously restored. Updates of the forwarding tables limit the steered transmissions to existing stations (3).

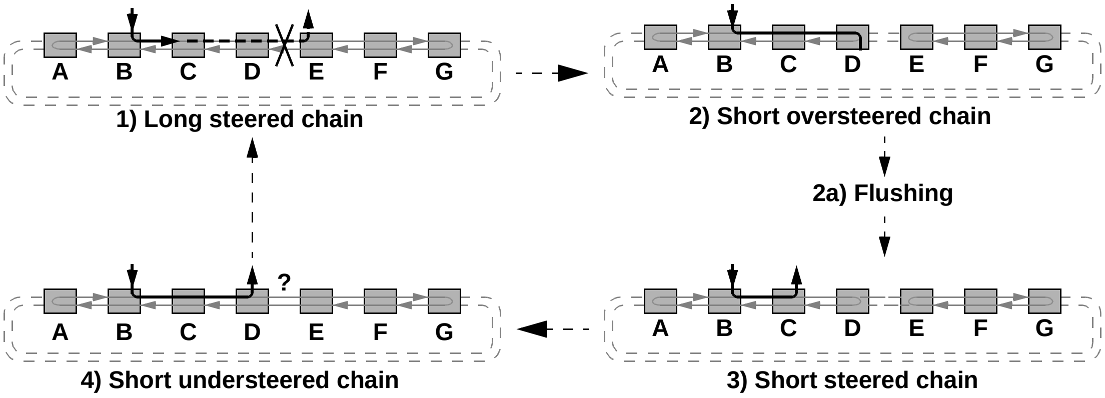


Figure F.29—Split and joined chains

The addition (4) of an additional chain makes all stations once-again visible, as illustrated in Figure F.29. Updates of the forwarding tables extend the steered transmissions to support reattached stations, fully restoring (1) the previously long-steered-chain topology. No flushing is necessary to support extended chains, since the curent forwarding table is a strict subset of the extended-topology forwarding table.

F.3.4 Flush management

A ringlet flush involves three sequential steps, listed below. To reduce MAC complexity and increase client flexibility, these operations are performed at the client, not the MAC.

a) Stopping client transmissions
b) Sending a flush frame to one’s self
c) Resuming client transmissions.

The client has the option to consider whether mis-ordering and duplication can be optionally allowed (like 802.1w), when flows are explicitly known to explicity such reorderings. With such bimodal strategies, the client can support ordered-flow requirements (which suffer an additional flush-circulation delay) without delaying other unordered flows.

NOTE—Flushing after flooding would theoretically allow the use of shortest-path remote-unicast fowarding, after unidirectional flooding, but such an approach suffers from additional latency, delay jitter, and serialization complexities.