>BUSINESS MADE **SIMPLE**

802.1aq Link State Protocol
SPBB Multicast Loop Prevention

A proposal

Dave Allan, Nigel Bragg, Jerome Chiabaut, Don Fedyk

NORTEL

Preface

- We have been looking for a scalable means to apply loop prevention to Multicast Trees
- The reason for protecting Multicast is that while data plane ingress checking (aka Reverse path forwarding check) protect from single failures of links. Multiple failures caused by single or multiple events are not completely protected. Replication of multicast traffic therefore need more stringent treatment.
- The following is a proposal for SPBB with data plane ingress checking.

Assumptions

> Loop prevention is required for multicast

> Loop mitigation applied for both multicast and unicast.

> We use A combination of source rooted shortest path trees to create a complete multicast group (*,G) becomes (S_1,G), (S_2,G)….etc

> Given the above, our approach is to remove multicast filtering data base entries when the path to the source of he multicast tree changes.

> Synchronization at an IS-IS Database level preferred for its message efficiency over per tree synchronization methods (but per database synch is a summary indication for all affected trees.)

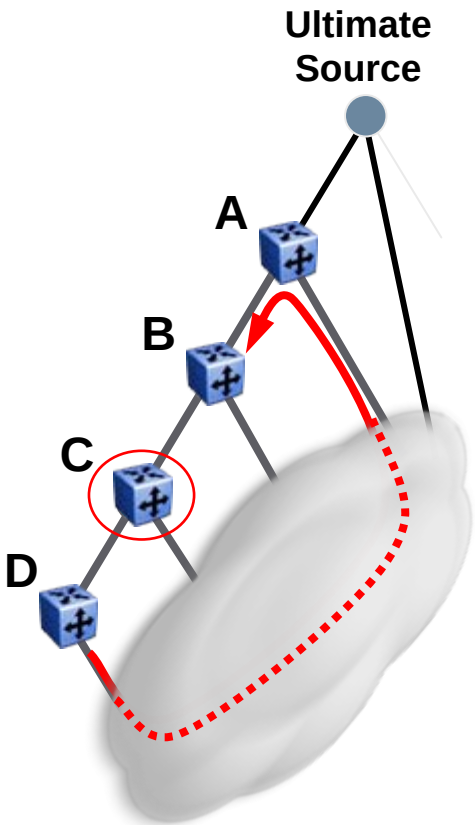
IS-IS Synchronization Approach is….

- If a node receives an LSP that changes the path to a root (the source), the node depopulates all FDB state for multicast from that root at the same time it updates the unicast entries
  - Note 1: We said path, not hop…!
  - Note 2: Unicast connectivity is restored quickly
  - Note 3: A single Dijkstra at the Node can be used to determine all affected multicast paths (due to congruency of trees).
- The node will only update previously depopulated multicast FDB state on interfaces when the node has synchronized with each new upstream neighbor
  - The neighbor has advertised that it has similarly depopulated state for affected multicast entries and updated the unicast entries
  - This protocol aspect is new but suggests a simple message such as a digest of LSP state.
- Result is multicast connectivity between a point in the network and a given root will be restored as soon as all nodes along the path agree on the path to the root
  - For some closer to the root, it may not have changed

Why does this work?

- Multicast state for a tree with a given root will not exist in a link between a pair of nodes unless both agree on the path to the root
  - If there was a moment of disagreement, they removed all state until they agreed again
    - Which is what occurs when an LSP causes a change in the path to the root
  - If they always agreed then the state did not change
    - A given LSP did not change that view for either one
- A complete path to the root is the logical “AND” of all pairs of nodes along the path agreeing
  - Else there is a discontinuity, and no FDB state for that root is populated at discontinuities
- RPFC similarly is a pair wise agreement but cannot not completely prevent loops
  - It still adds value, but is not an authoritative solution by itself
- The difference is that unlike RPFC, we depopulate all old multicast state caused by lack of control plane synchronization from the FDB
  - Farkas loop example depended on old multicast state
  - No old multicast state exists at an unsynchronized boundary

Detail…



We consider the conditions under which a node (**C**) can participate in a forwarding loop :

> Node **C** must be unaware of any topology change affecting its route to the source (else it would depopulate affected multicast state).

> other nodes must cooperate, in response to a topology change, to extract traffic from below **C** and return it to the tree above **C.**

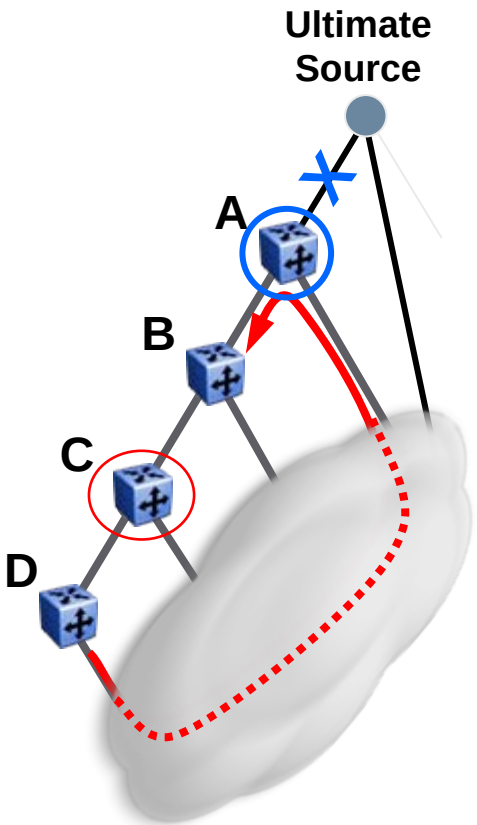
> If node **A** is able to forward as shown, then :

- **A** is aware of topology change(s) which have moved the shortest path to the source,
- and **A** has synchronised LSPs with its neighbours,

> but if node **C** is unaware of changes, there must be a topology knowledge discontinuity somewhere on its “current” route to the source :

- here node **B**, not synchronised with **C**, so blocking

Digging Deeper…



We need to address whether there is any race hazard in the event of multiple topology changes

> a **necessary** (but under RPFC not sufficient) condition for loop formation is for the shortest path to the source to change :

- if it does not, the topology of that tree does not change, all are in sync and converged wrt that tree

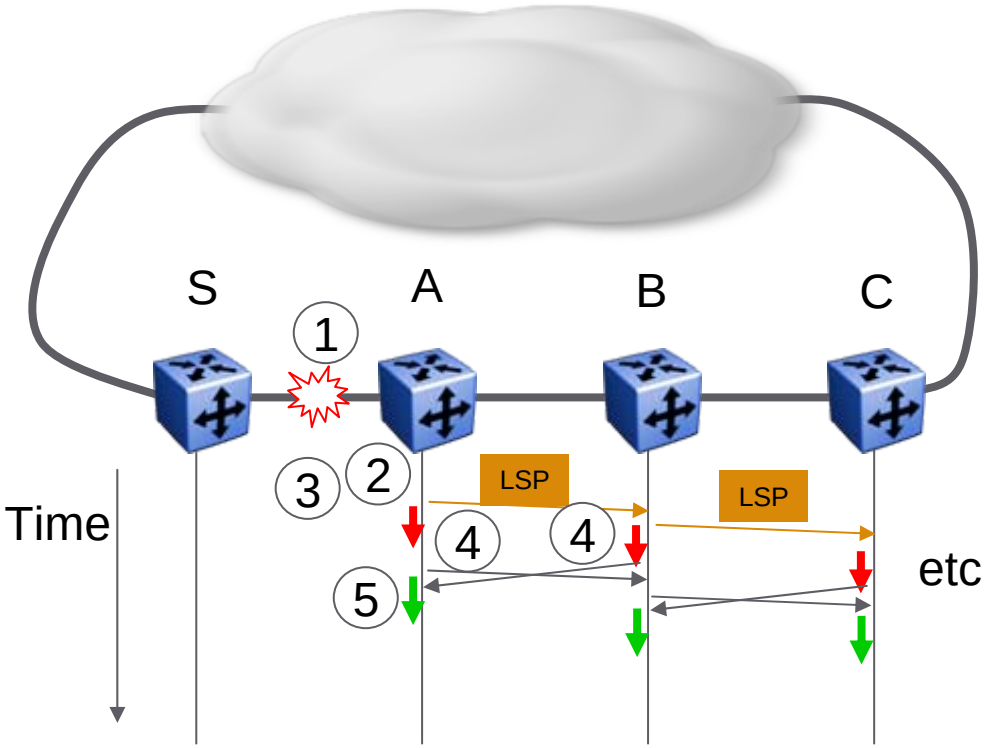
> the key node is therefore **A**, which must have received all the changes for it to decide that its SPF route to the source has changed :

- else **A** will still perceive the old route to be valid.

> As node **C** must be unaware of any topology change affecting its route to the source, there is a sync discontinuity, either between **A** and **B** (when **A** blocks), or between **C** and **B** (**B** blocks)

This argument applies to all nodes and all trees

Time line messaging

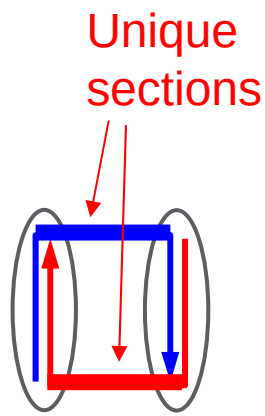


Change of Source Tree from S (viewed from A's perspective)

1. Fault
2. LSP Propagation
3. Removal of Multicast Source affected trees
4. Synch Messages
5. Safe Reinstall of Multicast originally sourced via S now sourced via B

Minimum Complexity of a loop with RPFC *Revisited*

- The minimum number of simultaneous topology changes to produce a loop with RPFC & consistent tie breaking is two
- If an individual node is not allowed to have two different views of the network simultaneously, a loop produced by two topology changes will have at least four nodes in it
- Why?
  - Two or more paths must be stitched together to form a loop
    - Without loss of generality, assume two, non-looping, paths
  - These paths can't agree all the way around the loop
    - There must be a break in each of them since they are acyclic
  - There must be section of the loop which is unique to each path
    - This section closes the loop when added to the other path
  - These sections must be joined by common sections:
    - section of path1 → common section → section of path2 → common section
  - The two common sections are distinct and each has a node at each end



The unique sections are artifacts that will be removed any time the path to the root has changed

If the common sections are synchronized, then two paths cannot simultaneously exist

Summary

- Explicit depopulation of multicast FDB state when a topology change alters the path to the root, combined with synchronized repopulation of updated multicast FDB state offers authoritative loop avoidance
- This is achieved without the delays that ordered convergence would bring
  - Synchronization handshaking only needs to reach to a point where the path to the root has not changed, not all the way to the root in order to restore multicast connectivity
- This is achieved without impacting unaffected multicast forwarding
  - If a path to a root unchanged, connectivity is maintained
- This does not impact unicast convergence