Route optimization

Current status

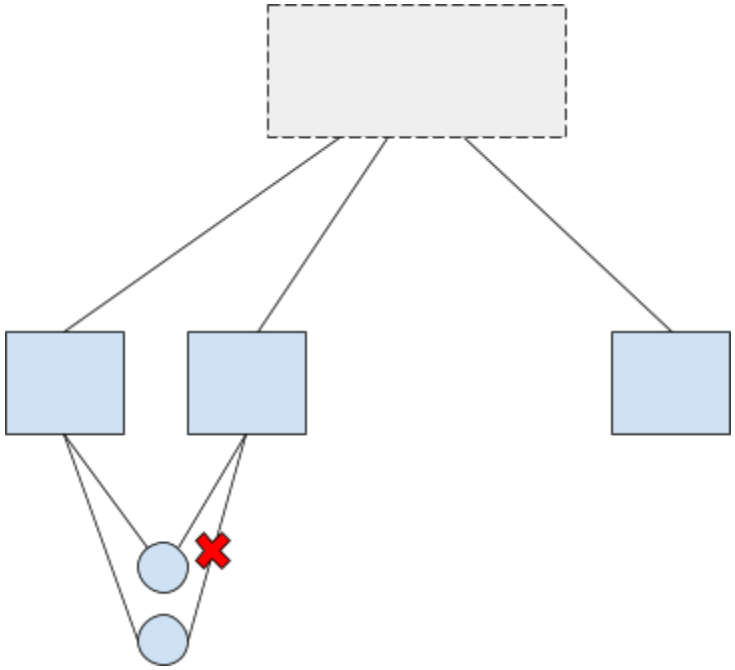
- **RouteAdded**
  - First time added, route is programmed in the Fabric. Flows+L3 unicast groups are added in the dst switch (*populateRoute*) and Flows+L3 ECMP groups are added in the rest of the fabric (*populateSubnet*).
- **RouteRemoved**
  - Route is definitely removed from the Fabric. Flows are removed in the dst switch (*revokeRoute*) and in the rest of the fabric (*revokeSubnet*).
- **RouteUpdated**
  - Route has been changed (new nexthops/new locations). Flows+L3 unicast groups are added in the new dst switch (*populateRoute*) and Flows are updated and new L3 ECMP groups are added in the rest of the fabric (*populateSubnet*).
- **Nexthop 2->1 location**
  - Route is the same, lost a location for the nexthop. Flows are removed from the old dst switch (*revokeRoute*) and Flows are added/updated and new L3 ECMP groups are added in the rest of the fabric (*populateSubnet*).
- **Nexthop 1->2 location**
  - Route is the same, added a location for the nexthop. Flows+L3 unicast groups are added in the new dst switch (*populateRoute*) and Flows are updated and new L3 ECMP groups are added in the rest of the fabric (*populateSubnet*).
- **LinkDown, LinkUp_seen_before, SwitchDown**
  - All ECMP groups are modified (and for switch down flows are eventually refreshed)
- **LinkUp_!seen_before**
  - Complete rerouting is executed. ECMP groups are modified and new Flows are installed

Problem

Operations related to nexthop changes or new links are very expensive. Working at scale means reinstall all the xxxK flows.

Current logic uses destination switches and hash together all the routes sharing the destination switches. From one side this allows to reduce the number of used ECMP groups but on the other side does not allow to efficiently fix issue or perform updates.

Let's consider **Nexthop 2->1 location**, we need to fix only the routes having this next hop. Current logic picks the affected routes, creates a new ECMP group and point the affected routes to this new ECMP groups.



Proposed solution

Firstly, let's not consider **LinkUp_!seen_before**, this is not a real problem.

In order to perform efficient operations we need to use more ECMP groups and at the same time have a more fine grained way to distinguish them.

DestinationSet store has to be augmented with the nexthop mac, ie mac1 and mac2. We are assuming that at most we support two different nexthops.

A new method has to be introduced in the *DefaultRoutingHandler* to heal the damaged routes or update the existing ones.

Any modification has to take into account current logic of **LinkDown, LinkUp_seen_before, SwitchDown**. This requires to revisit the logic handling these events (probably changes are minimal but we need to be careful).