# GroupFlow Flow Replacement Overview

## Flow Replacement Overview:

The GroupFlow module implements multicast routing for software defined networks using the POX network controller. This is implemented by having the module manage a map of MulticastPath objects, each of which is responsible for multicast route calculation and flow installation for a single, unique pair of multicast group IP address and multicast sender IP address. Multicast routes are generated by calculating shortest path trees using Djikstra's algorithm.

Flow replacement in this context will refer to the complete process of calculating and installing new routes for a particular pair of multicast group IP address and multicast sender IP address, which is implemented by the following algorithm:

1. The MulticastPath object retrieves a list of all links in the network from the GroupFlowManager object. This list consists of pairs of switch datapath identifiers.
2. For each link in the retrieved list:
   1. The MulticastPath object queries the FlowTracker module to determine the current utilization on the specified link.
   2. The current utilization on the link is used to determine a link weight for the specified link. The exact function used to convert the utilization to a link weight is determined by the **link\_weight\_type** configuration option.
   3. A tuple consisting of <switch DPID 1, switch DPID 2, link weight> is generated, and this tuple is added to a list of weighted edges.
3. The MulticastPath object now contains a complete list of weighted edges, which defines all links present in the network and their associated link weights .
4. The list of edge tuples is used to generate a set of all nodes (switch DPIDs) in the network. These two data structures together define a complete node-edge graph of the network.
5. Djikstra's algorithm is run on the network graph, using the switch attached to the multicast sender IP address as the source node. This operation produces a map which contains linked lists defining the shortest path to all nodes in the network from the source node.
6. The MulticastPath object queries its cache of desired reception state (provided by the IGMPManager) to determine which switches and output ports require multicast delivery for this group / sender.
7. For each combination of switch and output port, the shortest route to the specified switch is selected from the shortest path map, and Openflow commands are generated to enable delivery along the specified links / output ports. These Openflow commands will be OFPFC\_MODIFY or OFPFC\_ADD commands depending on whether flows for this multicast group / sender are already installed on the switch. All flows for a particular multicast group / sender are installed with the same, unique flow cookie (an arbitrary integer value).
8. Openflow OFPFC\_REMOVE messages are generated for any switches on which flows were previously installed for this group / sender, but are no longer required.
9. Openflow OFPFC\_MODIFY and OFPFC\_ADD commands are delivered to the network.
10. Openflow OFPFC\_REMOVE commands are delivered to the network.
11. The MulticastPath object caches a list of all installed rules, so they can be modified or removed as appropriate the next time flow replacement occurs.

## Flow Replacement Modes:

The GroupFlow multicast routing module currently supports three flow replacement modes. These modes determine the conditions under which flow replacement is triggered, but they do not impact the actual operation of flow replacement (i.e. the algorithm provided in the previous section is always followed, regardless of flow replacement mode).

### No Flow Replacement

Enabled by configuration option: **--flow\_replacement\_mode** **none**

In this mode, flows for a particular multicast group are only replaced only in the following cases:

* **The multicast group/sender is first initialized**: This only occurs when the controller first detects a valid pair of sender and receiver on the same multicast group address.
* **The multicast group state changes:** This occurs when receivers join or leave the multicast group. If the sender of the group leaves the network, or if all receivers in the group leave the network, all flows associated with the pair of multicast group/sender are removed from the network.
* **The network topology changes:** This occurs when a link goes up/down, or when a switch connects/disconnects from the network. In the current implementation, all multicast routes for all groups/senders are recalculated when a topology change occurs.

The mode can be considered as the minimum baseline flow replacement mode, as flow replacement must occur under these conditions to implement functionally correct multicast delivery. The conditions listed here will also trigger flow replacement in all other supported flow replacement modes.

### Periodic Flow Replacement

Enabled by configuration option: **--flow\_replacement\_mode periodic  
Options:**  
**--flow\_replacement\_interval** (specified in seconds)

In this mode, active flows are replaced at a periodic interval, as specified by the following algorithm:

1. When a multicast group/sender is initialized (i.e. when the controller first detects a valid pair of sender and receiver on the same multicast address) a periodic timer is started with an interval set by the **flow\_replacement\_interval** configuration option.
2. Flows are replaced each time this interval elapses, regardless of any replacement that occurred due to other conditions.
3. If the group sender leaves the network, or if all group receivers leave the network, the timer is stopped and associated resources are released.

For example, consider the following scenario:

* Periodic flow replacement is enabled, with a flow replacement interval of 10 seconds.
* A multicast group is initialized at time 0. Flow replacement occurs at this time, and a periodic timer is started with an interval of 10 seconds.
* At time 4 seconds, a new multicast receiver joins the group. This triggers flow replacement, according to the baseline conditions specified in the "No Flow Replacement" section.
* Periodic flow replacement occurs at time 10 seconds (despite replacement having occurred at 4 seconds), and every 10 seconds thereafter.

### Congestion Threshold Based Flow Replacement

Enabled by configuration option: **--flow\_replacement\_mode cong\_threshold  
Options:**  
**--flow\_replacement\_interval** (specified in seconds)  
**--cong\_threshold** (specified in Mbps)

In this mode, flow replacement is triggered by the FlowTracker module reporting congestion on a link. Upon receiving a LinkUtilizationEvent, the GroupFlow module will attempt to replace the largest flows traversing the link until the link is brought back under its congestion threshold. In this mode, the flow replacement interval option specifies a minimum interval between two replacements of the same multicast group/sender. This replacement mode operates according to the following algorithm:

1. The FlowTracker module polls all switches in the network at a periodic interval to determine the current bandwidth utilization of all inter-switch links in the network.
2. If the bandwidth utilization on a particular link exceeds the value set by the **cong\_threshold** option, the FlowTracker generates a LinkUtilizationEvent which is processed by the GroupFlow module.
3. The GroupFlow module calculates the desired replacement bandwidth as:  
    desired replacement bandwidth = current link bandwidth - congestion threshold  
   This value represents the amount of bandwidth which the GroupFlow module will attempt to replace in response to this particular event. The GroupFlow module effectively attempts to replace as many flows as are required to drop the utilization on the particular link under its congestion threshold.
4. The replacement bandwidth, representing the amount of bandwidth which has been replaced in response to this event, is set to 0.
5. The FlowTracker module reads the LinkUtilizationEvent to obtain a list of all flow cookies which are present on the specified link, as well as the bandwidth utilization associated with each cookie. This list is sorted in order of descending bandwidth (resulting in larger flows being replaced before smaller flows).
6. For each flow cookie in the list:
   1. The time at which the flows corresponding to the flow cookie were last replaced is compared with the flow replacement interval. If the flow has already been replaced within the specified interval, the loop continues without replacing the particular flow.
   2. If the flow has not been recently replaced, the multicast group/sender corresponding to the flow cookie has its flows replaced, and the estimated bandwidth usage of the associated flow cookie is added the replacement bandwidth. The last replacement time of the multicast group/sender is updated to the current time.
   3. If the replacement bandwidth meets or exceeds the desired replacement bandwidth, the loop breaks.

Two experimental options are available for this flow replacement mode, both of which may help reduce flow "thrashing" (i.e. overly frequent replacements).

* **Single Flow Replacement Mode:** In this mode, only a single multicast group/sender will have its flows replaced in response to each LinkUtilizationEvent. The loop specified by step 6 breaks as soon as a single flow is replaced (i.e. the condition in step 6.3 is not checked). The intuition behind this option is that replacing one flow at a time will achieve the same goal as the base algorithm (lowering the link's usage below its congestion threshold, if possible), but the change in traffic state will be more gradual, resulting in better bandwidth estimations.
* **Count Recently Moved Flows as Replaced:** In this mode,flows which are not actually replaced due to having already been recently replaced (i.e. fail the condition in step 6.1) are counted toward the replacement utilization. The intuition behind this option is that flows which were recently replaced to another link will still report bandwidth utilization on their previously used links for several polling intervals of the FlowTracker, as the FlowTracker averages its bandwidth estimates over several polling intervals.