Week13

**HyperFlow Controller Application**

**Initialization:** Upon NOX startup, the HyperFlow application starts the WheelFS client and storage services, subscribes to the network’s data and control channels, and starts to periodically advertise itself in the control channel. The advertisement interval must be larger than the highest round-trip time among controllers in a network. The advertisement message contains information about the controller including the identifiers of the switches it directly controls.

**Publishing events:** The HyperFlow application captures all the NOX built-in events (OpenFlow message events) as well as the events that applications register with HyperFlow. Then, it selectively serializes (using the Boost serialization library) and publishes the ones which are locally generated and affect the controller state. For that, applications must be instrumented to tag the events which affect their state. Furthermore, applications should identify the parent event of any non- built-in event they fire. This way, HyperFlow can trace each high-level event back to the underlying lower-level event and propagate it instead. Using this method we ensure that the number of events propagated is bounded by the number of the OpenFlow message events generated by the local controller.

**Redirecting commands targeted to a non-local switch:** A controller can only program the switches connected directly to it. To program a switch not under direct control of the controller, the HyperFlow application intercepts when an OpenFlow message is about to be sent to such switches and publishes the command to the network control channel. The name of the published message shows that it is a command and also contains the source controller identifier, the target switch identifier, and the local command identifier (similar to the event message identifier).

**Proxying OpenFlow messages and replies:** The HyperFlow application picks up command messages targeted to a switch under its control (identified in the message name) and sends them to the target switch. To route the replies back to the source controller, the HyperFlow application keeps a mapping between the message transaction identifiers (xid) and the source controller identifiers. The HyperFlow application examines the xid OpenFlow message events locally generated by the controller. If the xid of an event is found in the xid- controller map, the event is stopped from being further processed and is published to the network data channel. The name of the message contains both controller identifiers. The original source controller picks up and replays the event upon receipt.

**Health checking:** The HyperFlow application listens for the controller advertisements in the network control channel. If a controller does not re-advertise itself for three advertisement intervals, it is assumed to have failed. The HyperFlow application fires a switch leave event for every switch that was connected to the failed controller. Upon controller failure, HyperFlow configures the switches associated with the failed controller to connect to another controller. Alternatively, either nearby controllers can serve as a hot standby for each other to take over the IP address.