First Checkpoint Report

“FlowQoS: Not every flow is the same”

November 3, 2015

Group #20

Dhruv Sharma (A53101868)

Feichao Qian (A53091625)

Frederik Nygaard (U07002090)

Robert Jenkins (A53063069)

Completed Tasks

1. We've managed to setup the OVS connections on the assigned VM.
2. We have successfully been able to build the libprotoident library that serves as the core component of the traffic classifier module in FlowQoS SDN controller
3. We have the SDN controller (POX) setup on the VM to control the OVS pair using the FlowQoS classifier module.
4. Tested the DASH-JS player to confirm that it can be used to measure the bitrate.
5. Started browsing through the FlowQoS control logic to understand the packet events and corresponding actions it takes to classify them as one of the available traffic types.

Pending Tasks for the Next Checkpoint

1. Run experiments using adaptive bitrate video streaming using Dash-JS player with and without the use of FlowQoS traffic classifier, to attain the comparison between the adaptive bitrate streaming performance as well as throughput for either cases. While we are waiting for a logistic request regarding the VM networking, this task might get delayed by a few days, but we intend to cover-up during the next week tasks.
2. We understand the use of Linux Traffic Control and meaning of classless and classful qdiscs.
3. Experiment with it on the egress traffic to see how we can mark the packets based on their traffic types and use Hierarchical Token Bucket (HTB) and the classes we associate with it.
4. We write the end host agent to execute commands to mark packets based on the classification done by the FlowQoS controller.

Graphs

1. Behaviour of adaptive bitrate-based video streaming with and without FlowQoS classification. Using this graph we aim to show that in the presence of heavy background traffic, video streaming tends to adapt to a poor bitrate in order reduce the delay and data loss. Using FlowQoS we expect to see a relatively high and stable bitrate across time due to traffic isolation that it provides. The y-axis will be the bitrate of the video streaming application (in Kbps) whereas the x-axis will be time of the experiment (in secs).
2. Performance of adaptive bitrate-based video streaming with and without FlowQoS classification. Using this graph we aim to show that in the presence of heavy background traffic, video streaming adapt to low throughput to reduce the data loss in the process. Using FlowQoS we expect to see a relatively higher throughput across time due to traffic isolation that it provides. The y-axis will be the throughput (in Kbps) whereas the x-axis will be time (in secs).
3. Since, VoIP traffic is highly sensitive to network, we experiment with a VoIP call in the presence of background TCP traffic with and without FlowQoS. We expect to see lower round-trip time (RTT) values along with low deviation from the mean with FlowQoS than without it due to the isolation it provides. The y-axis will be the round trip time (RTT) (in msecs) whereas the x-axis will be time (in secs).
4. Also, a VoIP call requires the video and audio to be as smooth as possible, which in-turn requires the jitter for VoIP traffic to be low. We use low rate iperf UDP traffic to measure the packet jitter experienced with and without FlowQoS. The y-axis will be the jitter (in msecs) whereas the x-axis will be time (in secs).
5. While static and isolated queues as provided by FlowQoS does indeed improve the performance of sensitive traffic types, it doesn't use the available bandwidth optimally. We experct that our traffic classes using Linux Traffic Control can overcome this issue without any change in the quality of service that FlowQoS provides. The y-axis will be throughput for iperf TCP and adaptive bit-rate video traffic (in Kbps), whereas the x-axis will be time (in secs).