QoS based Optimal Route Discovery

using OpenFlow

Problem Description

Routing algorithms traditionally route packets using greedy approach to datagram delivery. Algorithms are broadly classified into Distance Vector and Link State algorithms. The primary metrics used by these algorithms to calculate shortest path is number of hops or the link-cost defined based on the bandwidth. Essentially when a new shortest path is identified, all the traffic is routed in that direction. Nowhere in this technique do we take the QoS requirements of the traffic into consideration. There may exist cases where a new path is calculated and the traffic is routed to it, even though the old path could still be satisfying the QoS requirements.

In our project we design a routing scheme based on the QoS requirements of the traffic.

We take into consideration link latency, bandwidth and queue/port drops as the QoS metrics to calculate the shortest path based on the service requirement(ToS/DSCP). These metrics are fed to a cost function based on which the shortest path is calculated based on the service requirements.

Components

Platforms for Project

**OpenFlow**: It is a Software Defined Networking (SDN) paradigm that provides centralized control functions by decoupling control and data forwarding layers of routing.

**Mininet**: Mininet creates a **realistic virtual network**, running **real kernel,switch and application code**, on a single machine.

**POX**: It is an OpenFlow controller development framework in Python. It provides various openow APIs that can be used to interact with switches, fetch statistical information from the switches, and write control information to the switches.

**Open vSwitch**: Open vSwitch is a production quality, multilayer virtual switch licensed under the open source [Apache 2.0](http://www.apache.org/licenses/LICENSE-2.0.html) license.  It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols.

Major Components



Figure (1) : Block Diagram