Latency-sensitive Flow management in a Software-Defined Network

Riccardo Mutschlechner, Stephen Sturdevant, Kausik Subramanian

March 14, 2016

1 Measurement Model

In software defined networks, the network is managed by a central controller and switches can be programmed by the OpenFlow API by the controller. OpenFlow also supports queries to switches, which would retrieve statistics from the switches. For this project, we can assume a measurement model where each switch is probed every t_m seconds (the measurement interval) time.

Work on this to do presently :

- Figure out the different statistics supported by OpenFlow. Primarily, we would need switch queue sizes, and flow statistics on the switch.
- Build a preliminary monitoring system using POX and Mininet.

2 Estimating Flow Characteristics

Something similar to CSFQ, ideally we would want to predict the throughput of a flow with respect to time (denoted by Θ). Look at the next section for decisions based on this. This would use the measurement model and t_m and other factors to estimate the throughput function for the flow. An assumption we can make in this case is that a flow (identified by a set of headers) will occur regularly in the datacenter. This is not unreasonable, consider distributed applications. For example, short queries can occur between different modules (will possess the same flow headers), thus estimating a flow can reap benefits in the future as we can make better decisions for this particular flow. Another assumption we can consider is that the throughput function is periodic (?) or based on some distribution. This needs to be thought about.

3 Flow Decisions

3.1 Queue buildup model

Flows in datacenters require strong latency guarantees, and one of the major factors causing latency is queue buildup in switches. Thus, it is of paramount importance to be able to quantify queue buildups in switches to make better scheduling decisions. By building a queue buildup model, we aim to achieve two things: (1) Provide latency guarantees (2) Predict congestion. Using the queue buildup model, we can predict when latency guarantees or congestion will happen in the future, and accordingly decide which flows to move around.

For a switch sw_i and output port j, let us define F_i^j to the be the set of flows traversing through the switch to switch j and the link has capacity C_{ij}. The Queue buildup function Ω_i^j is as follows :

$$\Omega_i^j(t) = \Omega_i^j(0) + \Sigma_{f \in F_i^j} \int_0^t \Theta_i^f(t) - C_{ij}t$$

The other consideration we need to make is the modifications to Θ as the throughput will vary depending on the utilization on each switch the flow traverses through. The modified throughput function is defined as follows :

$$\Theta_j^f(t) = min(C_{ij} \frac{\int_0^{\Gamma_i^f} \Theta_i^f(t)dt}{\Sigma_{f \in F_i^j} \int_0^{\Gamma_i^f} \Theta_i^f(t)dt}, \Theta_i^f(t))$$

The modified active time interval of the flow Γ_j^f is defined as follows :

$$\Gamma_j^f = \frac{\int_0^{\Gamma_i^f} \Theta_i^f(t)dt}{\Theta_j^f(t)}$$

The modified inactive interval of the flow $\overline{\Gamma_j^f}$ is as follows :

$$\overline{\Gamma_j^f} = \overline{\Gamma_i^f} - (\Gamma_j^f - \Gamma_i^f)$$

3.2 Flow Durations

By building a model of the flow, we can extract two important times for improving decision-making in SDNs. We assume flows follow a ON/OFF model.

- t_{on} : The duration of time the flow is active. While this can help classify elephant and mice flows; in SDNs there is another use case for this. The rules installed by the controllers have a timeout value after which the rule is discarded. By estimating accurate time durations, we can set the timeout values so that the rules are removed after the flow becomes inactive, thus increasing the efficiency of switch tables.

- t_{off} : The duration of time when the flow is OFF. This can be used by the controller to pre-emptively install rules. By doing this, the latency- sensitive flows do not suffer the additional overhead of the controller adding the rules on the switches.