Probabilistic Controller Selection in Software-Defined Networks

10\textsuperscript{th} August 2013

Minsu Shin (djscape@korea.ac.kr)
MNC Lab.
Korea University
Contents

• Introduction
• Probabilistic Controller Selection
• Preliminary Result
• Conclusion
Introduction

• Software-defined networks (SDNs) is an emerging technology

• Recently, large-scale SDNs have much attention
  – SDN-enabled Internet
  – Cellular networks

• Multiple controllers should be deployed in large-scale environments
Probabilistic Controller Selection (PCS)

• Objectives
  – Network-wide load balancing and flow-specific flow latency performance improving

• Operation
  1. A switch receives the first packet of a flow
  2. It selects its serving controller by computing the controller selection probability
  3. The switch forwards the first packet to selected controller
Controller Selection Probability (1/2)

• Notation
  – $C$: # of controllers
  – $p_i^*$: selection probability of controller $i$
  – $p_i^N$: selection probability of controller $i$ considering network-wide load balancing
  – $p_i^S$: selection probability of controller $i$ considering flow-specific latency performance
Controller Selection Probability (2/2)

• \( p_i^* = \alpha p_i^S + (1 - \alpha) p_i^N \), \((0 \leq \alpha \leq 1)\)

• The sum of \( p_i^* \), \( p_i^S \), and \( p_i^N \) are 1

• \( \alpha \) is weight parameter
  – If the network-wide load balancing is more important, \( \alpha \) close to 0
  – Otherwise, \( \alpha \) close to 1
Two controllers example based on M/M/1 queuing system

Poisson arrival process

Flow arrival rate ($\lambda$)

First packets of flows

$RTT_1$

$\lambda p_1^*$

Controller 1

Service rate of Controller 1 ($\mu_1$)

$\mu_2$

$RTT_2$

$\lambda p_2^*$

Controller 2

Service rate of Controller 2 ($\mu_2$)

Exponential distribution
Network-wide load balancing probability

• The offered load to each controller should be balanced

\[
\frac{\lambda p_1^N}{\mu_1} = \frac{\lambda p_2^N}{\mu_2}
\]

\[
p_1^N = \frac{\mu_1}{\mu_1 + \mu_2}, \quad p_2^N = \frac{\mu_2}{\mu_1 + \mu_2}
\]

• \( C \) controllers generalization

\[
p_i^N = \frac{\mu_i}{\sum_{j=1}^{C} \mu_j}
\]
Flow-specific latency performance probability

- Minimizing the average flow setup latency ($T_S$)

$$T_S = \frac{(p_1^S)^2}{\mu_1 - \lambda p_1^S} + \frac{(p_2^S)^2}{\mu_2 - \lambda p_2^S} + p_1^SRTT_1 + p_2^SRTT_2$$

$$[p_1^S, p_2^S] = \arg \min_{p_i^S \in [0,1]} (T_S)$$

- $C$ controllers generalization

$$p^S = \arg \min_{p_i^S \in [0,1]} (T_S)$$
Preliminary Result (1/2)

- Network-wide load balancing performance
  - PCS can provide well-balanced offered loads regardless of disparity in service rates
  - When $\alpha$ is zero, perfect load balancing can be achieved

$\lambda = 2000$, $\mu_1 = 4000$, $\mu_2 = 4000 \sim 9000$ [flows/sec]
Preliminary Result (2/2)

- **Flow-specific latency performance**
  - PCS can effectively reduce the flow setup latency when RTT difference is large
  - When $\alpha$ is 1, flow setup latency can be reduced the most

- Apparently, *trade-off*

\[ \lambda = 2000 \]
\[ \mu_1 = 4000, \mu_2 = 3000 \text{ [flows/sec]} \]
\[ RTT_1 = 1\text{ms}, RTT_2 = 1\text{~}6\text{ms} \]
Conclusion

• In probabilistic controller selection (PCS), a serving controller is selected probabilistically.
• Both the network-wide load balancing and flow-specific latency performance can be achieved.
• There is a trade-off, and thus $\alpha$ should be adjusted depending on the environment.