LazyCtrl: Scalable Network Control for Cloud Data Centers

Kai Zheng*, Lin Wang+, **Baohua Yang***, Yi Sun+, Yue Zhang*, Steve Uhlig#

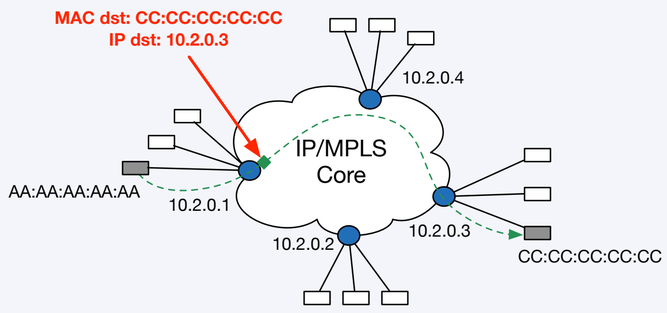
*IBM Research +Chinese Academy of Sciences

#Queen Mary University of London

MOTIVATION

A. Data Center Networking

- Location resolution
- Address mapping



B. Traffic Locality in Cloud Data Centers

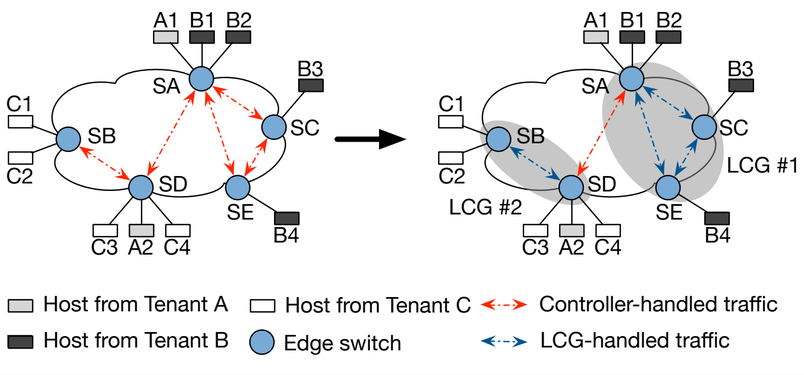
- The traffic distribution is **uneven** among hosts (90% of the flows are contributed by about 10% of the host pairs)
- The traffic appears to be **concentrated** within some groups of hosts (the centrality of grouping 6509 hosts into 5 groups is 0.853)

Inter-tenant traffic is small & tenant sizes are relatively stable

SYSTEM DESIGN

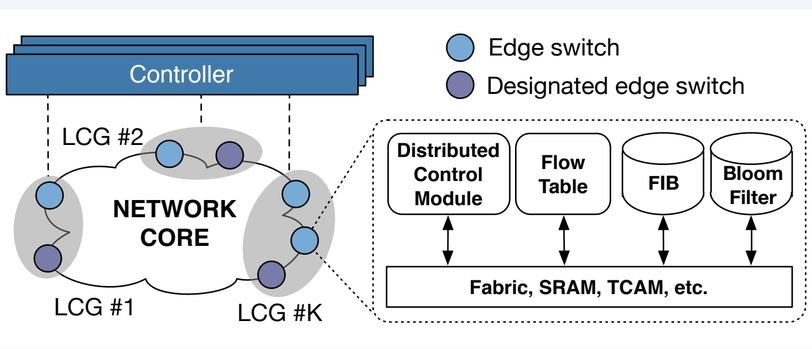
A. Design Overview

The edge switches in a data center are clustered into multiple switch groups according to traffic affinity. Those switch groups are called Local Control Groups (LCGs). The controller clusters the switches in such a way that the size of each group is maximized under a given limit while the inter-group traffic volume is minimized to support its laziness. The following figure depicts a simple example of the design.



B. Architecture

In the architecture design of LazyCtrl, the network is separated into two parts: the *core* and the *edge*. The network adopts a hybrid control model where the control plane consists of a logically centralized controller and distributed control modules in the LCGs.



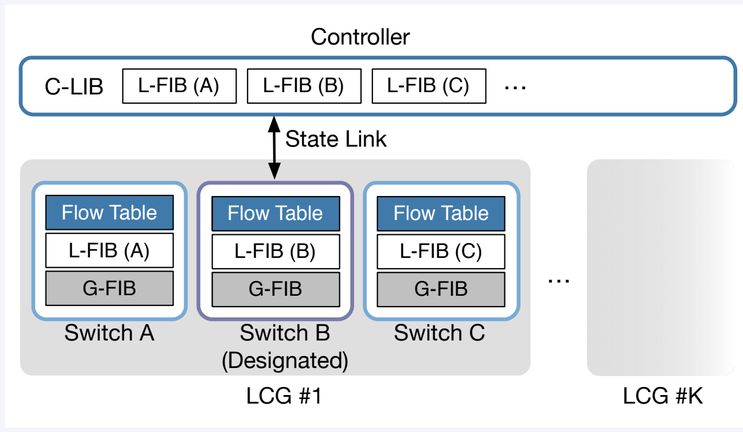
C. Switch Grouping

The edge switches are grouped according to traffic affinity. For a given switch grouping scheme, we define the inter-group traffic intensity by

$$W_{\text{inter}} = \sum_{\{x,y\in[1,\ldots,K]\wedge x\neq y\}} \sum_{\{S_m\in G_x, S_n\in G_y\}} w_{mn}.$$

The objective is to minimize the inter-group traffic intensity. We base our design on a Multi-Level k-way Partition (MLkP) algorithm. To mitigate the time overhead of frequent grouping update, we also provide an incremental update process for fast regrouping.

D. Table Organization



- **L-FIB:** Local Forwarding Information Base, used for local hosts that are attached to the switch
- **G-FIB:** Group Forwarding Information Base, used for hosts in the same LCG, implemented with Bloom Filter
- **Flow Table:** containing rules generated by the controller
- **C-LIB:** Central Location Information Base, including the L-FIBs from all the edge switches

E. Failover

We provide a wheel-wise failure detection mechanism and handle both link and switch failures.

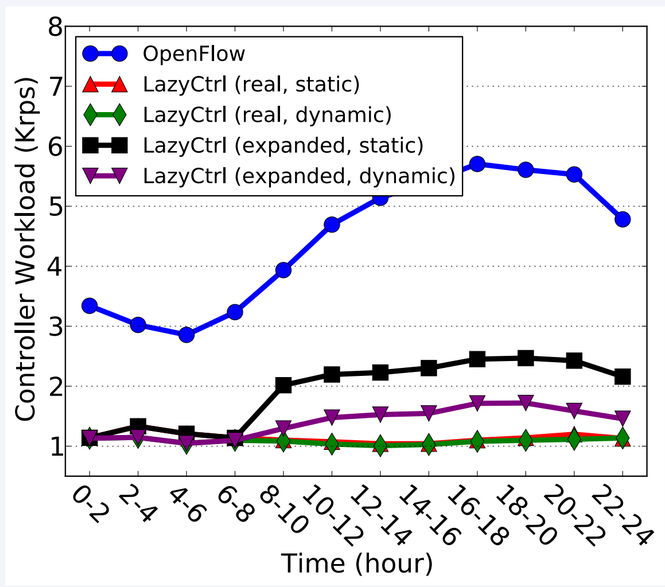
Failure	Packet loss		
	$S_n \rightarrow S_{n-1}$	$S_n \rightarrow S_{n+1}$	Controller$\rightarrow S_n$
Control link			✓
Peer link (Up)	✓		
Peer link (Down)		✓	
Switch (S_n)	✓	✓	✓

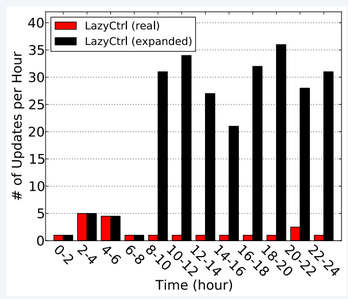
EXPERIMENTS

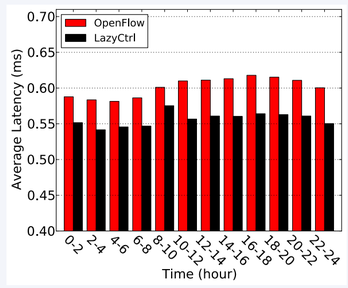
A. Implementation

Network Core
6 Pronto 3290 switches
10 GigE links
Network Edge
24 IBM x3550 8-core servers
272 Linux virtual hosts (with the modified OVS)
Controller
Linux PC (Intel Core 2 Duo CPU 2.2 GHz) with the modified Floodlight controller

B. Testbed Evaluation







LazyCtrl

1) can largely reduce the workload of the controller
2) needs to update infrequently
3) can reduce the latency compared to purely centralized solutions

The 35th International Conference on Distributed Computing Systems (ICDCS 2015)