Advanced Computer Networks Report

Lab3: Building A Data Center Network

1. Team members (Group 10)

- Corneliu-Dumitru Soficu [2675454]
- Andrei-Ioan Boloș [2701860]

2. Experiment Setup

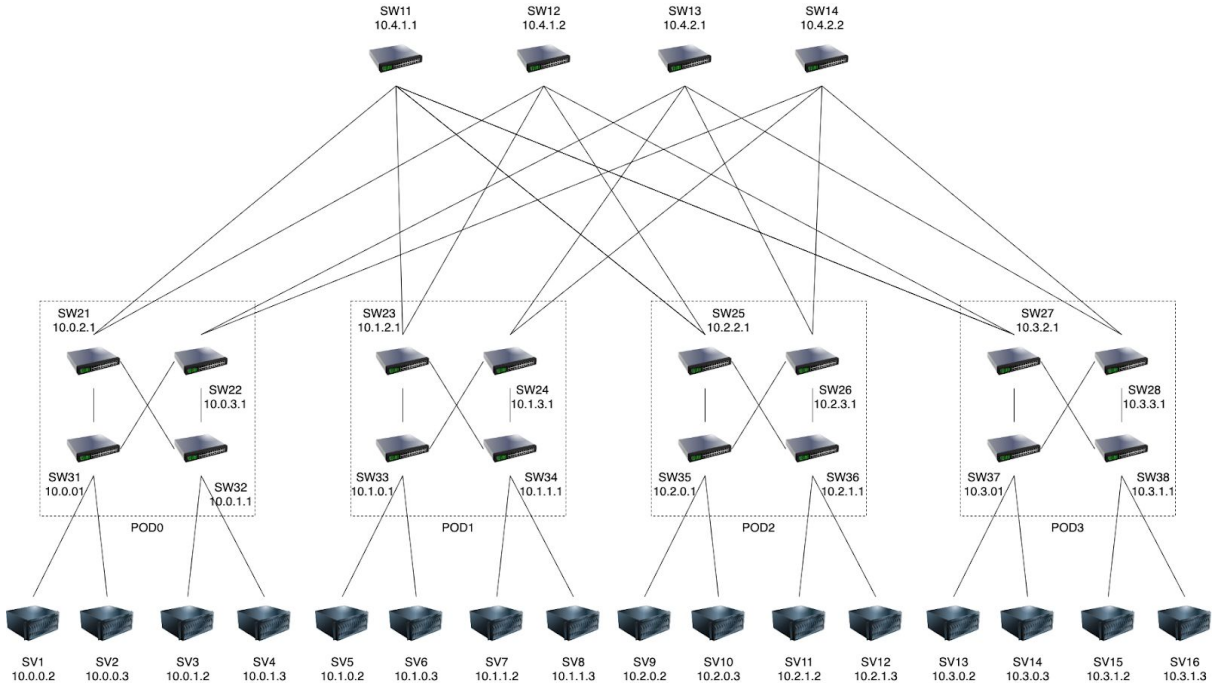

Fig. 1 - Fat-Tree Topology (k = 4)

For this experiment, we are using the k=4 Fat-Tree topology (Fig 1). We have tested the performance, measuring latency using ***ping***, and throughput using ***iperf***.

We have named all servers using the SV prefix followed by the server index, starting from 1 to 16: SV1, SV2, … SV16. SV1 is the bottom leftmost server and incrementally counting we reach SV16 which is the rightmost bottom host.

For core switches, we have used the SW1 prefix followed by the core switch index: SW11, SW12, SW13, SW14.

For aggregation switches, we have used the SW2 prefix followed by the switch index: SW21, SW22, SW23 … SW28.

For edge switches we have used the SW3 prefix followed by the switch index: SW31, SW32 … SW38.

We have configured every link with a network speed of **15 Mbits/sec** and a delay of **5ms**.

Source Pod	Source	Destination	Destination Pod
Pod 0	SV1 (10.0.0.2)	SV5 (10.1.0.2)	Pod 1
Pod 0	SV4 (10.0.1.3)	SV6 (10.1.0.3)	Pod 1
Pod 2	SV9 (10.2.0.2)	SV13 (10.3.0.2)	Pod 3
Pod 2	SV11 (10.2.1.2)	SV14 (10.3.0.3)	Pod 3

*Table 1: Experiment configuration for simultaneous **iperf** connections*

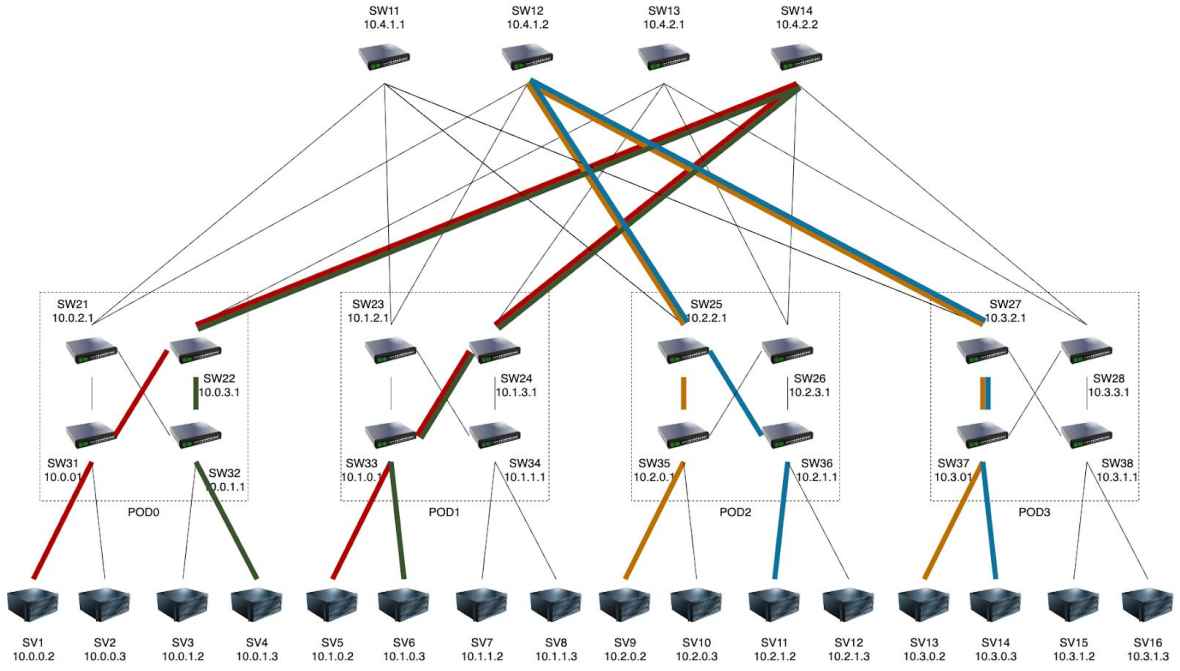

Fig. 2 - Dijkstra Routing for the interpod traffic

Before executing our experiments, we executed the **pingall** command in order to allow all switches to learn the flows to all servers, depending on the routing strategy tested: Dijkstra/Two Level.

To test the throughput, we've decided to use a scenario for interpod communication where POD0 communicates with POD1 while at the same time, POD2 communicates with POD3. For this we've chosen the pairs of servers from *Table 1* and ran the **iperf** command for 1 minute from source to destination for all pairs of servers simultaneously.

For the **Dijkstra Routing case** (Fig. 2), the algorithm chose the following 6 HOP paths for all the server pairs:

1. POD0 -> POD1:
    a. **SV1** -> SW31 -> SW22 -> SW14 -> SW24 -> SW33 -> **SV5**
    b. **SV4** -> SW32 -> SW22 -> SW14 -> SW24 -> SW33 -> **SV6**
2. POD2 -> POD3:
    a. **SV9** -> SW35 -> SW25 -> SW12 -> SW27 -> SW37 -> **SV13**
    b. **SV11** -> SW36 -> SW25 -> SW12 -> SW27 -> SW37 ->**SV14**

We can already notice that for the routes from POD0 to POD1 and from POD2 to POD3, 3 out of 6 links for both cases are shared by 2 connections. Because of this we expect the throughput to be half of the available 15Mbps capacity.

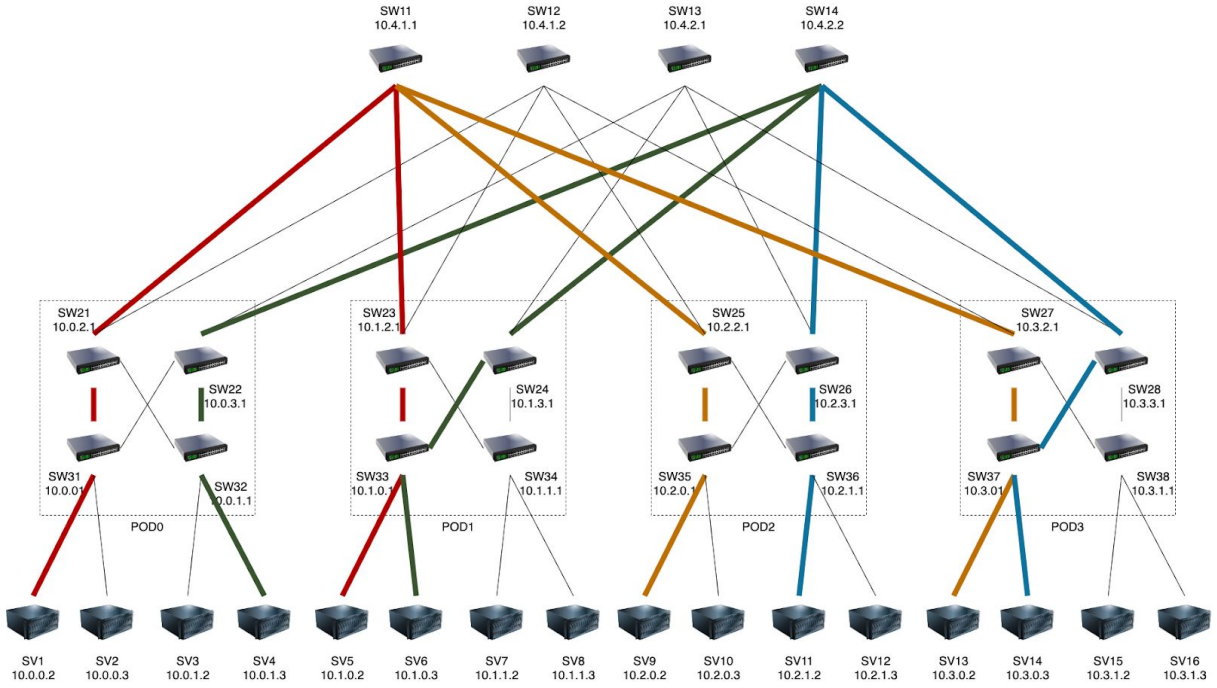

Fig. 3 - Two level routing for the interpod traffic

For the routing using **Two Level Routing Tables** (Fig. 3) the algorithm chose the following 6 HOP paths for all the server pairs:

1. POD0->POD1:
    a. **SV1** -> SW31 -> SW21 -> SW11 -> SW23 -> SW33 -> **SV5**
    b. **SV4** -> SW32 -> SW22 -> SW14 -> SW24 -> SW33 -> **SV6**
2. POD2 -> POD3:
    a. **SV9** -> SW35 -> SW25 -> SW11 -> SW27 -> SW37 -> **SV13**
    b. **SV11** -> SW36 -> SW26 -> SW14 -> SW28 -> SW37 -> **SV14**

What we can already notice for the Two level routing case is that no links are shared by simultaneous connections. This should have the advantage of maximizing the throughput for all connections.

3. Results and Conclusions

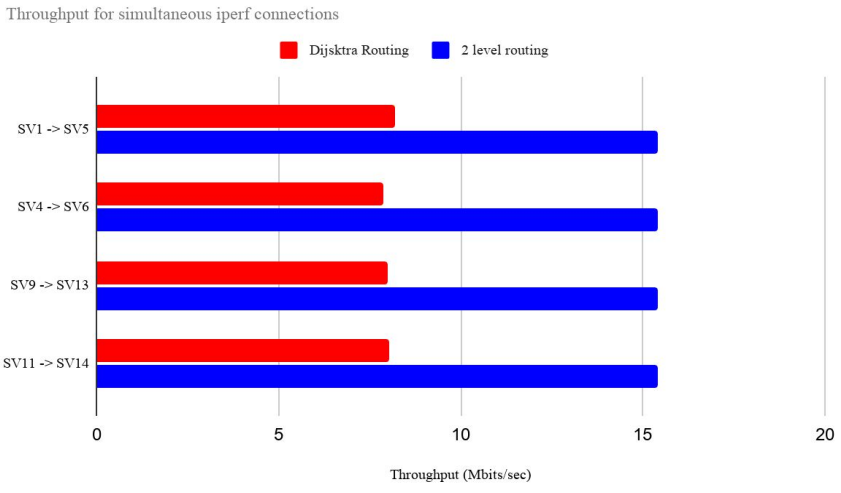

*Fig. 4 - **iperf** results for both routing cases*

Fig.4 shows the performance difference when running multiple simultaneous **iperf** connections for 60 seconds. The figure clearly displays the performance difference between both routing strategies.

For the **Dijkstra algorithm**, the achieved throughput for all server pairs is around **7.8 Mbits/sec**. This is due to the fact that connections for all server pairs have to be shared. Because the Dijkstra algorithm does not implement any load balancing techniques, it will always pick the same link when connecting two pods, even when there are multiple connections from different servers from the same pod.

The **Two Level algorithm** managed to achieve the maximum throughput in the network, interpod connections achieving a consistent throughput of **15.4 Mbits/sec**. The reason for this is because all connections were routed through distinct routes resulting in no shared links. This properly demonstrates the load balancing capabilities of the Two Level Algorithm. By looking at Fig. 3, we also notice that all ports of the SW11 and SW14 core switches are utilized compared to Fig. 2 where only half of the ports of the SW12 and SW14 core switches are utilized. This load balancing capability is mainly achieved by the second level suffix from the two level routing table which instructs an aggregation switch to send a packet destined for another pod to a different edge switch depending on the ending of the destination IP.