Core layer

IP: 10.8.i.j

共16个。

Aggregation layer

IP: 10.pod.switch.1 pod: 0~7 switch: 4~7

每个pod 4个。共32个。

Edge layer

IP: 10.pod.switch.1 pod: 0~7 switch: 0~3

每个pod 4个。共32个。

Host

IP: 10.pod.switch.ID pod: 0~7 switch: 0~3 ID:2~k/2+1(原文为2~k/2-1，应该是错的。比如k=4时，范围就是2~1了。)

每个pod 16个。共128个。

Routing table

# Edge layer

如匹配，这直接转发至host端。若不匹配，则根据IP地址最后一位，将转发请求分摊到同一pod的4个edge layer上。Destination为-1则直接投递，其余为下一跳IP地址。路由文件输出格式为每8行为一组，按顺序对应一个路由。0.0.0.0的情况不输出，如果前4行不匹配自动到第五行寻找子表。

Example：for 10.0.1.1

Prefix Output Port Destination

10.0.1.2/32 0 -1

10.0.1.3/32 1 -1

10.0.1.4/32 2 -1

10.0.1.5/32 3 -1

0.0.0.0/0 🡪

Suffix Output Port

0.0.0.2/8 4 10.0.4.1

0.0.0.3/8 5 10.0.5.1

0.0.0.4/8 6 10.0.6.1

0.0.0.5/8 7 10.0.7.1

# Aggregation Layer

和文档里一样。

路由文件输出格式为每8行为一组，按顺序对应一个路由。0.0.0.0的情况不输出，如果前4行不匹配自动到第五行寻找子表。

# Core layer

每个core layer 都会连接到每个pod的中一个edge layer。前4个core layer 连接 pod 中的第一个edge layer，第5~8，连接pod 中的第二个edge layer。。。。相连的对应关系为

10.8.i.j 🡪 10.pod.(4+i-1).1

没有子表。

Example: 10.8.2.3

Prefix Output Port Destination

10.0.0.0/16 0 10.0.5.1

10.1.0.0/16 1 10.1.5.1

10.2.0.0/16 2 10.2.5.1

10.3.0.0/16 3 10.3.5.1

10.4.0.0/16 4 10.4.5.1

10.5.0.0/16 5 10.5.5.1

10.6.0.0/16 6 10.6.5.1

10.7.0.0/16 7 10.7.5.1

10.0.1.2到10.2.0.3的路由过程为：

先找到0pod 1switch的edge layer，路由表为Edge\_Layer\_route第9行（0\*4\*8+1\*8+1）~17行开始，匹配0.0.0.3/8,5,10.0.5.1，转发到0pod 5switch 的 Aggregation layer。IP为10.0.5.1查找Aggregation\_layer\_route第9(0\*32+(5-4)\*8+1)行至17行，查找到0.0.0.3/8,3,10.8.2.2，转发到core\_layer, IP 10.8.2.2。查找core\_layer\_route。（2-1）\*4\*8+（2-1）行为该路由路由表起始位置，匹配到10.2.0.0/16,2,10.2.5.1，转发至Aggregation layer。Aggregation layer 路由表起始为73行(2\*32+(5-4)\*8+1)，匹配10.2,0.0,0/24,10.2.0.1。转发至edge layer，路由表起始为65行（2\*4\*8+0\*8+1）至65行，匹配到10.2.0.2/32,0,-1，发送成功。