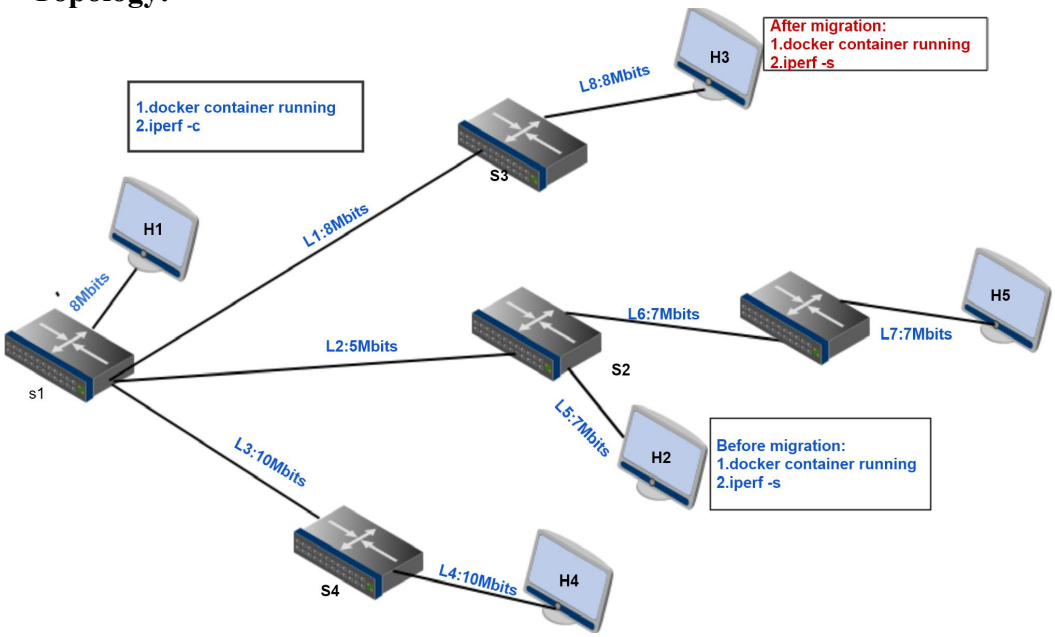
Report for December

Experiment of different policies

Goal: Policy based Docker Container migration in SDN environment, by collecting results and doing the plot, do the comparison and analyse different policies.

- **Topology:**



Why choose this topology:
L2 can be a bottleneck link for the bandwidth
L2 can be a factor of choosing the shortest path
It is also suitable for shortest path and random

Note:
All the migration destination is chosen in the free host set(no docker container running)

- **Use Cases**

Use case1:
Policy: bandwidth
Scenario: H1 H2 run Docker container which hosts iperf server service
Traffic generation: H1 as iperf client, H2 as iperf server
Migration Analysis: using **Max(min(bandwidth))** select one has the most available bandwidth

H2 is detected as heavy host(migration source host)
H2 -> H3 , path bandwidth=5
H2 -> H4 , path bandwidth=5
H2 -> H5, path bandwidth=7
So H5 is selected as migration destination host

Migration decision: H2->H5

Use case2:
Policy: shortest path
Scenario:H1 H2 run Docker container which hosts iperf server service
Traffic generation: H1 as iperf client, H2 as iperf server
Migration analysis: using floodlight REST API query the fastest path,according to the path latency

H2 is detected as heavy host(migration source host)
H2 -> H3 , path H2->S2->S1->S3->H3
H2 -> H4 , path h2->S2->S1->S4->H4
H2 -> H5 , path h2->S2->S5->H5
So H5 will be considered as the shortest path with least latency

Migration decision: H2 -> H5

Use Case3:
Policy: random
Scenario: H1 H2 run Docker container which hosts iperf server service.
Traffic generation: H1 as iperf client, H2 as iperf server.
Migration Analysis: randomly select migration source host from busy host set,
migration destination host from free host set.
busy host set (H1, H2)
Free host set (H3,H4,H5)
Migration decision: H1 -> H3 , H1 -> H4 , H1 -> H5
H2 -> H3 , H2 -> H4 , H2 -> H5

● Plots set

Where does these plots come from?
->For bandwidth consumption plot,the value is the sum of send and receive bandwidth consumption ,time starts before migration occur. It experiences migration process,and some time after migration done. The plot clearly showed the bandwidth consumption changes before, during,and after migration.
->For thoughput accumulation plot, the value is the number of bytes transfer along with the time. It showed that the totally traffic during the migration process.

Figure1(Use Case1)

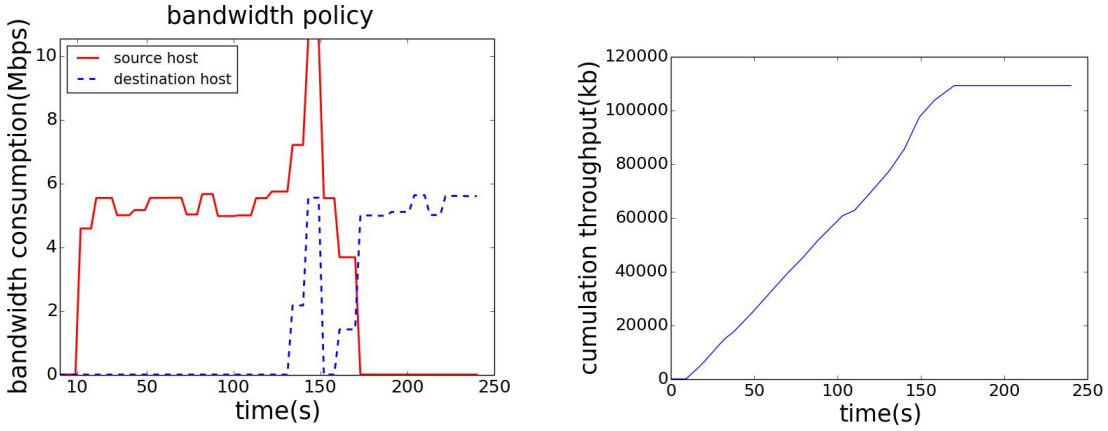


Figure2(Use Case2):

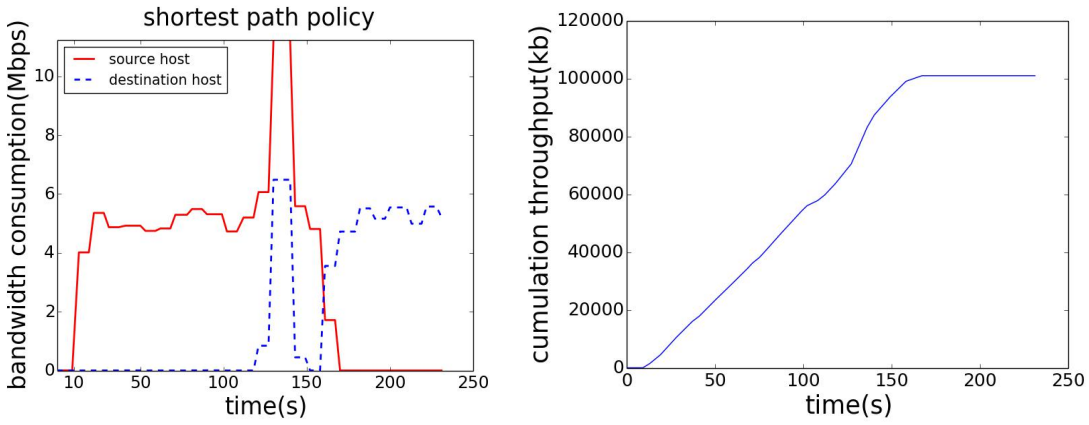
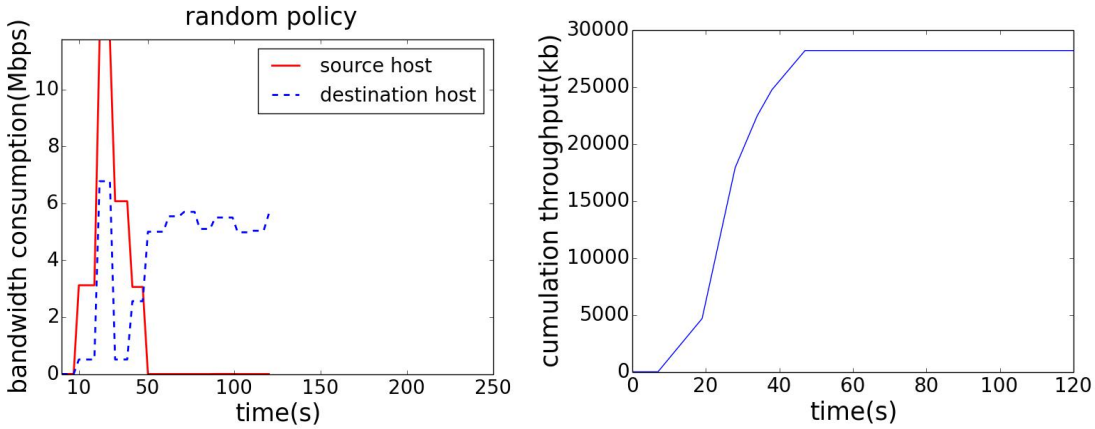


Figure3(Use Case3):



From those figures, we can conclude that all the migration policy have the same change shape.The difference is the traffic quantity and the time when it starts the migration. Analyzing those figures, we can characterize 3 period:

First period, the traffic data is collected before migration process from the system start running, so for the red line(source host) initially it just running the docker container which acts as iperf server, it receive and send data to iperf client.

second period, as the time going after first period, the iperf traffic generating is increasing. Then the switch source host attached is detected as heavy switch and the host is selected to be the migration source host. Then, It starts the migration process. During this period, the source host not only generates traffic with the iperf client, but also has the traffic with the migration destination host for docker image file transfer. As for the blue line(destination host), it start to receive the migration file, so bandwidth consumption is starting increase during this interval. Then after migration is done, it enters the third period.

Third period, the red line(source host), no more running the iperf server,so there is no traffic anyone(since here we only have iperf server,no other server). Instead of

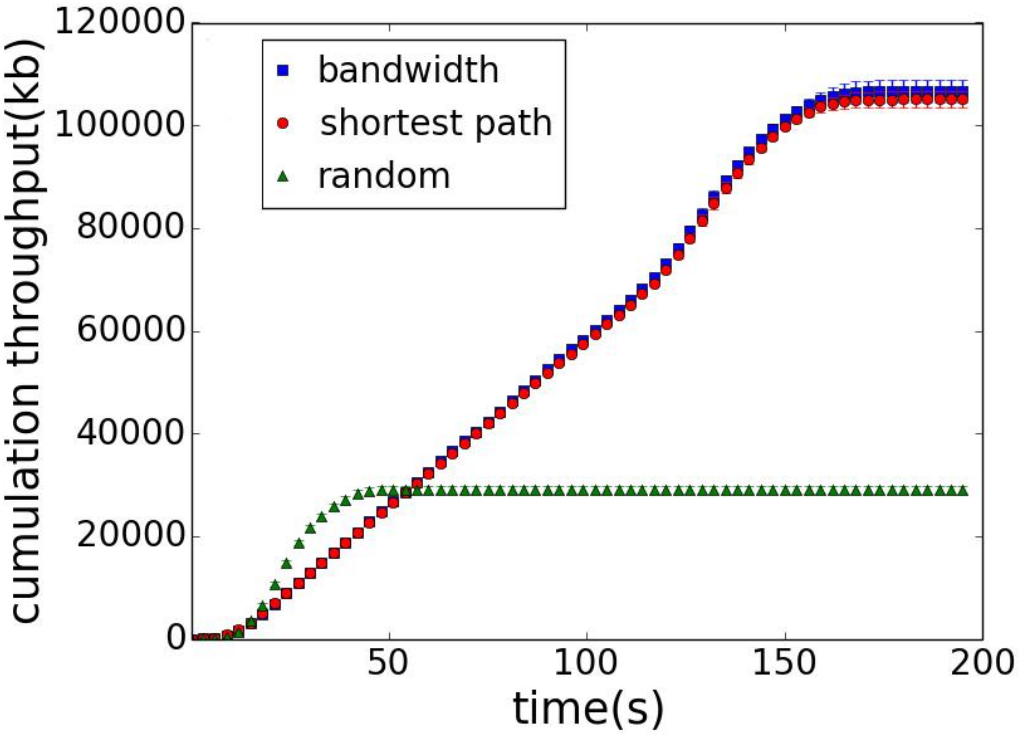
blue line(destination host), it starts to run the iperf server after migration, it will have traffic from the iperf client,So the bandwidth consumption for source host is increasing continuously.

- **Thoughput Accumulation Comparison**

Where does these data come from?

The traffic is collected from migration source host before migration process starting. It stops when the migration is done, here the host only runs iperf server, so when the server is migrated from the host, there is more traffic anymore, the cumulation throughput enter a stable stage.

Figure4(cumulation throughput combination)



Vertical comparison:

It shows that in the first 50s, the throughput accumulation of random policy is increasing faster than bandwidth and shortest path policy. It implies that system starts migration process earlier when using random policy than bandwidth and shortest path policy.when random migration occurs, the network situation is not too bad, and the migration source host doesn't suffer a high load of traffic. After 50s, shortest path already finished the migration process, the accumulation doesn't increase anymore. Instead of bandwidth and shortest path, it continue increasing, this means the network traffic is heavier and heavier until when it is detected, and migration is done. Bandwidth and shortest path has larger number, it means that when migration occurs,

the migration source host is suffering heavy traffic load. There is much more traffic in the network.

Horizon comparison:

Here we compare the saturation point, which means the migration process complete time. It can be seen clearly that random policy arrives much earlier than bandwidth and shortest path policy. When the network has high requirement of latency, for example, migration must be done in 50s, it is better to choose random policy.

- **Migration Time Comparison**

Table1(migration time 10.0.0.2-10.0.0.5)

	bandwidth	shortest path	random
1	12s	12s	12s
2	12s	37s	12s
3	12s	13s	12s
4	14s	17s	12s
5	15s	17s	12s
6	15s	28s	12s
7	15s	17s	12s
8	17s	28s	12s
9	37s	19s	12s
10	18s	21s	12s
11	33s	19s	12s
12	18s	20s	12s
13	31s	19s	12s
14	18s	20s	12s
15	29s	19s	13s
16	19s	20s	13s
17	27s	20s	13s
18	24s	20s	13s
19	25s	20s	13s
20	25s	20s	13s
average	(20.8 ±2.7)s	(20.3 ±1.9) s	(12.3 ±0.2) s

Table2(migration time random policy based)

	10.0.0.2-10.0.0.3	10.0.0.2-10.0.0.4	10.0.0.2-10.0.0.5
1	30s	29s	12s
2	33s	34s	12s
3	37s	37s	12s
4	31s	29s	12s
5	35s	35s	12s
6	38s	38s	12s
7	31s	32s	12s
8	35s	35s	12s
9	38s	38s	12s
10	31s	32s	12s
11	35s	36s	12s
12	43s	39s	12s
13	32s	33s	12s
14	35s	36s	12s
15	32s	34s	13s
16	36s	36s	13s
17	32s	34s	13s
18	37s	36s	13s
19	33s	34s	13s
20	37s	37s	13s
average	(34.6 ±1.2) s	(34.7 +1.0) s	(12.3+0.2) s

From table1, it can be seen that the migration time between bandwidth, shortest path, random,from the same source host 10.0.0.2 to the same destination host 10.0.0.5, concerning the average migration time, random has the smallest migration time. Shortest path,and bandwidth they are almost the same.
Behind story: For bandwidth and shortest path policy, the main goal is load balancing,the system first detect the heavy switch, then apply the policy algorithm, so in this case, the available bandwidth for migration will be much smaller, this results in larger migration time. Instead of random, it doesn't concern the load of each switch, just simply choose the source and destination. Most of the case, before the switch getting heavy, it starts the migration process,so
the available bandwidth is larger, this results in smaller migration time.

From table2, it can be seen that the migration time various under policy random while migrating to different host. It proves that L2 is the bottleneck link with small bandwidth. When migration to 10.0.03 or 10.0.0.4, it always needs larger migration time. It implies that random policy has uncertainty, this feature can be used to do simple attack defence. But in terms of the migration performance, it can not guarantee the best migration destination.

Combine table1 and table2, we can conclude that shortest path and bandwidth can do load balancing, after migration, it can guarantee the better service performance. But under switch overloading, migration process takes longer time. Random policy is good for attack defence, but it concerns less the traffic situation. It does migration even when switch is not overloaded.