Report for week4

Database update:

In the first stage, i use java collections List and Map to store the statistics.But in order to do the analysis more efficiently, Probably integrates with hadoop and spark applications to make more advanced decision choice. In the second stage i use xampp Apache server, move the data storage from the application storage to mysql database.
The table is following:

Connection:

source	dstType	destination
S1-eth1	host	h1-eth0
S1-eth1	server	s2-eth2

Explain:
Mapping the switch port to the host port in order to know which host need to migrate.
From the floodlight statistics, the information we can get is the switchDPID and port,so the connection table which get from mininet or from the configuration file from the network administrator. After from the statistics analysis, we can get the switch port which is overload,or satisfy the condition to migrate.Then from this table can find the corresponding host.

Mapping:

name	**value**
00:00:00:00:00:00:00:01_1	S1-eth1
h1	10.0.0.1

Explain:
This mapping table is match the ip address and host name, and the switchDPID and port with the name.The IP address is used to do the TCP connection.

StatisticAggregate:

switchDPID	**Time**	**Flow_count**	**Packet_count**	**PCDifference**
00:00:00:00:00:00:00:01	**2017-09-25 15:41:47**	**3**	**5750**	**734**

Explain:
The table can be as the data input of the intelligence decision algorithm, it can be used to measure the packets difference. If the packets difference change dramatically,it could have the high posibility that the switch port experiencing a heavy traffic.

StatisticsBandwidth:

switchDPID	**port**	**time**	**bitsPerSecondRx**	**bitsPerSecondTx**
00:00:00:00:00:00:00:01	1	2017-09-25 15:37:06	0	126

Explain:
This table could be the second data input of the intelligence decision algorithm,when find the switch Package aggregate increase dramatically,then can check the bandwidth,in which port that the bandwidth is the minimum,then the corresponding host could do the migrate.

Where and how to construct the VIB:

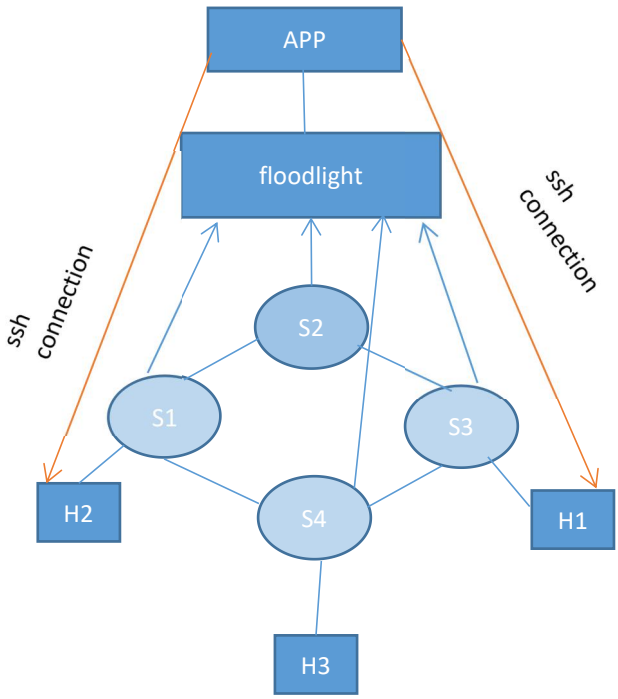
1. From mininet get the connections between host and switch ---> connection tables
2. From floodlight get host information and switch information do mapping,because floodlight use port number and DPID as the identifier of the switches. But mininet or in the real case, it is convenient to use the name to identify the switch and hosts. --->mapping table
3. From floodlight using the REST API, to get the information of the bandwidth and aggregate info --->StatisticsBandwidth,StatisticsAggregate table

Basic idea of the intelligence decision:

Firstly,select the top switch which has the highest difference change,and the port which had the minimum bandwidth. Then the corresponding host as the migration source.
Then from the bandwidth table,select the top 3 ports has the max bandwidth, the corresponding switches as the possible migration destination,send requests to floodlight to query the path latency.
Finally, select the path which has the smallest latency,the destination of the path is the decision migration destination.

Migration Architecture:

Option 1:



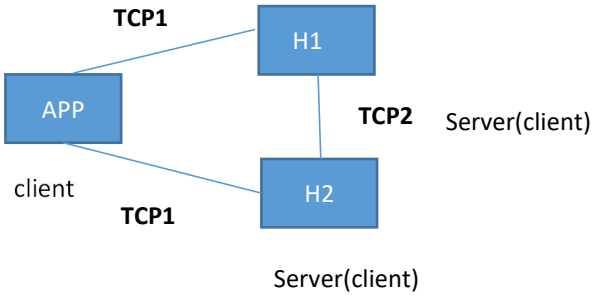
After the decision is made,the source and destination hosts is found.
Suppose:H1 is the source,and H2 is the destination.

1. The app open ssh connection access H1, execute docker command,to save the current status of the container.
2. Then the app execute the second command in H1, using the ssh connection. Do scp transfer the files that are got from the step1.
3. The app open another ssh connection with H2,then access H2 ,restart the container.

Pros:
The architecture is simple,host do nothing,everything is managed by the app.
Cons:
The app is heavy, all the migration task is done by the app,so the app should be more robust to tollerate the failures.

Option2:

The core architecture is the same with option1, but the communication way between app and hosts is different. Instead of using ssh connection, here using TCP connection.



Architecture roles:

- Each host run as a server,and app run as a client.
- host running the server process wait for the connection from: App or host.
- If server receive connection from APP,it will receive the command from the app do migration.
- If server receive the connection from host it will receive the files which need to do migration.

Basiclly there will be two programs,one is the client program,and another one is the server program.

Process:(suppose host 1 is the source and host 2 is the destination)

1. **Client(app) after make decision open TCP connection to Host1**
2. **Host1(server) check the command if it need to do migration or restart**

--- If the command is "containnerMigration", do docker check point,then as a client open a TCP connection to host2 for files transfer.

--- if the command is "containerRestart",prepare to receive the files, and execute container restart command.

Protocol:

1) Client(app) after make the decision,know the source and destination of IP address.
Open TCP connection to the source,send command:

"migration"
"destination host IP"

2) The Server(Host) listening TCP port 8088,waiting for commands from app:

pseudo code:

```
switch(command):
case   "containnerMigration":
       do docker check command;
       Open another tcp connection to destination host.
          Send the files which save the concurrent status of the container.
case   "ContainnerRestart":
          Receive files;
          Execute the command to restart the container;
```