

Project Status for Elf

Reporter: Ze Yu and Xiaolin Li

1. Current Project Status

The project is completed. We have built a fully-functional SDN service and integrated it with Hadoop (see the documentation for details). We have deployed in our local datacenter to test its functionality.

Some facts of this project:

Lines of code: ~3000 lines of Java code

New Java classes: ~40 new classes

2. Test results

We planned to test it on both local datacenter and ExoGeni testbed. Here are our testing results.

2.1 Local datacenter

2.1.1 Environments

We deployed our application on a local cluster consisting of 8 nodes. These nodes are distributed across 8 different racks. These racks are then connected into a tree-topology like most datacenter networks. Two racks form a datacenter. Thus, we have 4 datacenters in total. The topology is shown in Figure. 1 and devices are listed in Figure. 2.

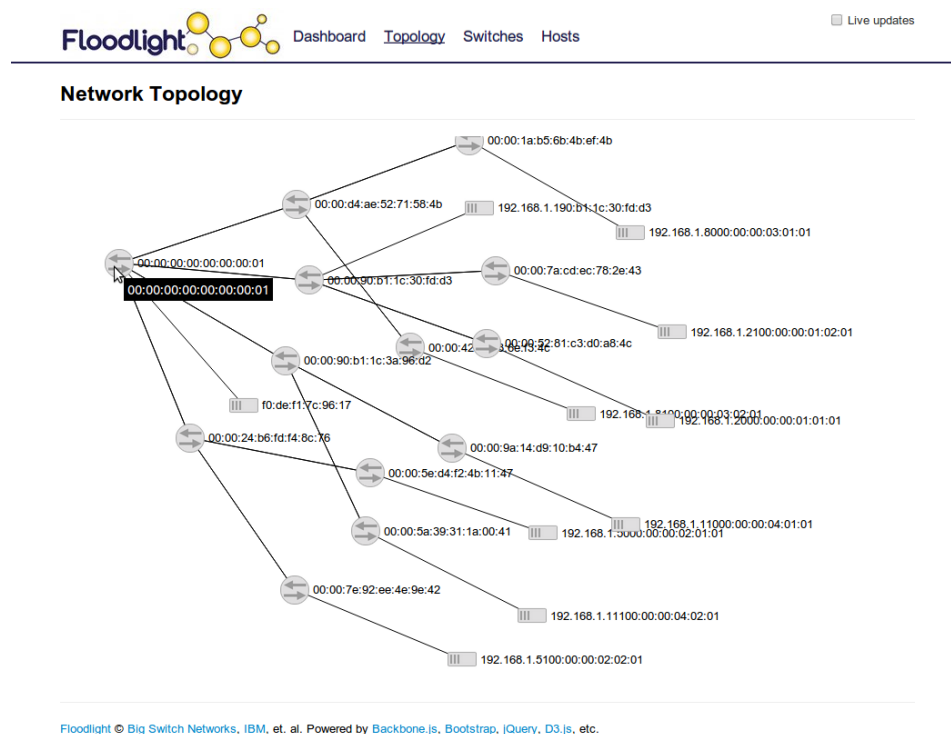


Figure 1. Local Testbed Topology

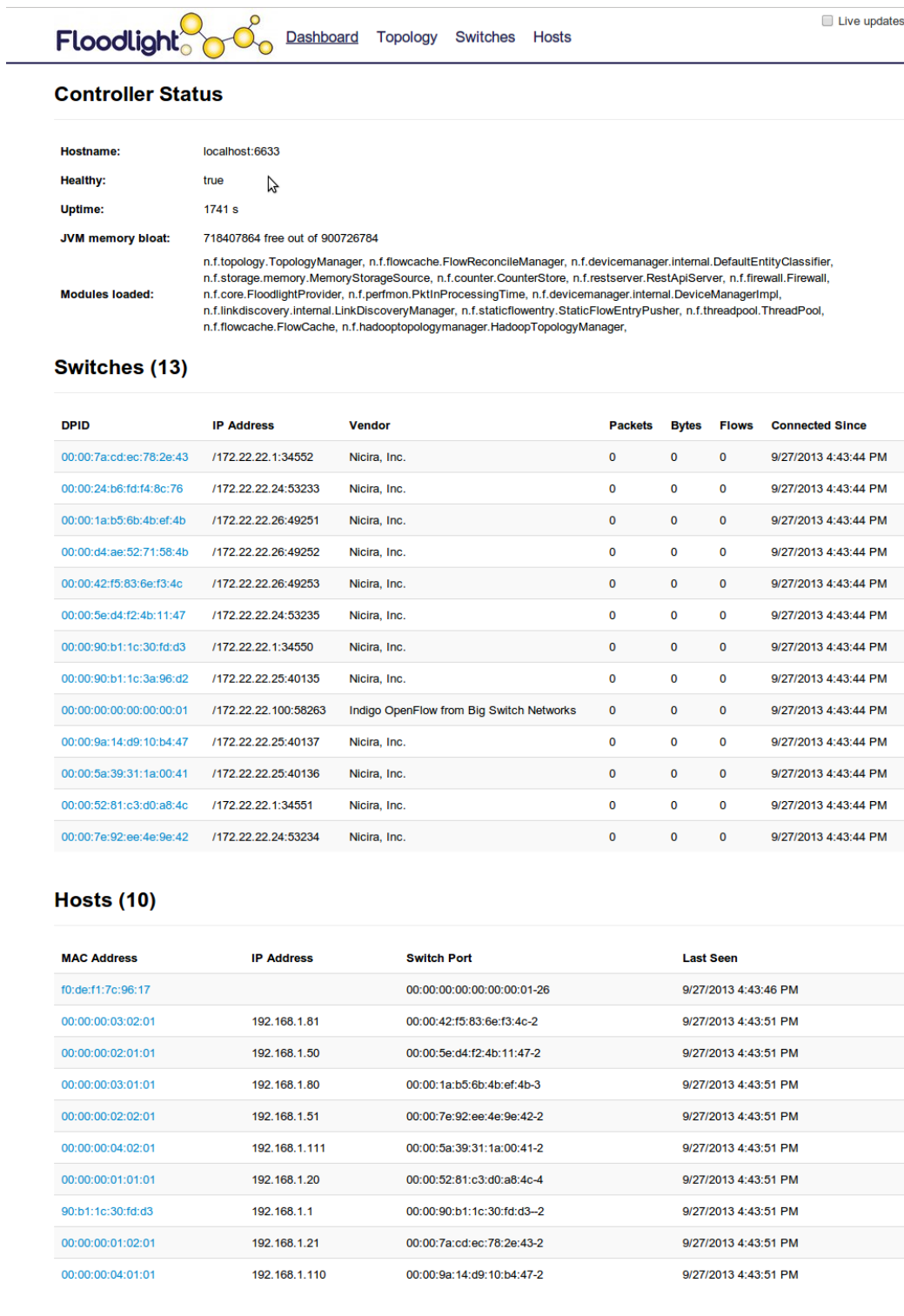


Figure 2. Device List

The nodes run in Ubuntu Server 13.04, with 3.2 Linux kernels. All codes are compiled using Sun JDK 1.6 and Apache Ant 1.8.

2.1.2 Test Results

We have tested both the SDN based topology management and the Hadoop integration. After setting up the testbed and configuring the network, we test the network Elf network topology management service by querying host locations via RESTful API that the Elf provides:

```
s3lab@headnode:~/elf$ curl -s http://10.227.119.160:8080/wm/hadooptopology/?host=192.168.1.50/00:00:00:00:00:00:01/dc2/00:00:5e:d4:f2:4b:11:47
```

Figure 3. Elf Service Query Result

We can see that given a host IP address, the Elf service returns its network location in the multi-datacenter topology.

Next we test the multi-datacenter aware optimization of Hadoop based on the network topology management service that Elf provides. First, we show that our datacenter-aware data placement works well. Our placement strategy now is to place 3 replicas in two different datacenters: one datacenter has two replicas on two different racks and the other has one more replica.

```
/user/ubuntu/input/b 329123080 bytes, 5 block(s): OK
0. blk_2292545193883778511_1001 len=67108864 repl=3 [/00:00:00:00:00:00:01/dc4/00:00:5a:39:31:1a:00:41/192.168.1.111:50010, /00:00:00:00:00:00:01/dc1/00:00:52:81:c3:d0:a8:4c/192.168.1.20:50010, /00:00:00:00:00:00:01/dc4/00:00:9a:14:d9:10:b4:47/192.168.1.110:50010]
1. blk_1941964895107307826_1001 len=67108864 repl=3 [/00:00:00:00:00:00:01/dc4/00:00:9a:14:d9:10:b4:47/192.168.1.110:50010, /00:00:00:00:00:00:01/dc4/00:00:5a:39:31:1a:00:41/192.168.1.111:50010, /00:00:00:00:00:00:01/dc1/00:00:52:81:c3:d0:a8:4c/192.168.1.20:50010]
2. blk_2005226386850894849_1001 len=67108864 repl=3 [/00:00:00:00:00:00:01/dc3/00:00:1a:b5:6b:4b:ef:4b/192.168.1.80:50010, /00:00:00:00:00:00:01/dc3/00:00:42:f5:83:6e:f3:4c/192.168.1.81:50010, /00:00:00:00:00:00:01/dc1/00:00:52:81:c3:d0:a8:4c/192.168.1.20:50010]
3. blk_5584102875604240394_1001 len=67108864 repl=3 [/00:00:00:00:00:00:01/dc2/00:00:7e:92:ee:4e:9e:42/192.168.1.51:50010, /00:00:00:00:00:00:01/dc2/00:00:5e:d4:f2:4b:11:47/192.168.1.50:50010, /00:00:00:00:00:00:01/dc1/00:00:52:81:c3:d0:a8:4c/192.168.1.20:50010]
4. blk_6095365225741781855_1001 len=60687624 repl=3 [/00:00:00:00:00:00:01/dc2/00:00:7e:92:ee:4e:9e:42/192.168.1.51:50010, /00:00:00:00:00:00:01/dc1/00:00:52:81:c3:d0:a8:4c/192.168.1.20:50010, /00:00:00:00:00:00:01/dc2/00:00:5e:d4:f2:4b:11:47/192.168.1.50:50010]
```

Figure 4. Datacenter-aware Block Placement Strategy

Finally, we run a Hadoop job under the datacenter-aware prefetching and scheduling policy:

```

ubuntu@vm-1-1:~/elf/hadoop$ ./wordcount
13/09/27 17:43:16 INFO hdfs.DFSClient: ==elf== prefetch client connecte
13/09/27 17:43:16 INFO hdfs.DFSClient: ==elf== prefetch client connecte
13/09/27 17:43:17 INFO input.FileInputFormat: Total input paths to proc
13/09/27 17:43:17 INFO mapred.JobClient: Running job: job_201309271742_
13/09/27 17:43:18 INFO mapred.JobClient: map 0% reduce 0%
13/09/27 17:43:35 INFO mapred.JobClient: map 3% reduce 0%
13/09/27 17:43:36 INFO mapred.JobClient: map 6% reduce 0%
13/09/27 17:43:37 INFO mapred.JobClient: map 13% reduce 0%
13/09/27 17:43:38 INFO mapred.JobClient: map 17% reduce 0%
13/09/27 17:43:39 INFO mapred.JobClient: map 22% reduce 0%
13/09/27 17:43:40 INFO mapred.JobClient: map 34% reduce 0%
13/09/27 17:43:41 INFO mapred.JobClient: map 45% reduce 0%
13/09/27 17:43:42 INFO mapred.JobClient: map 52% reduce 0%
13/09/27 17:43:43 INFO mapred.JobClient: map 60% reduce 0%
13/09/27 17:43:44 INFO mapred.JobClient: map 71% reduce 0%
13/09/27 17:43:45 INFO mapred.JobClient: map 76% reduce 0%
13/09/27 17:43:46 INFO mapred.JobClient: map 79% reduce 0%
13/09/27 17:43:50 INFO mapred.JobClient: map 82% reduce 0%
13/09/27 17:43:52 INFO mapred.JobClient: map 83% reduce 0%
13/09/27 17:43:53 INFO mapred.JobClient: map 85% reduce 0%
13/09/27 17:43:54 INFO mapred.JobClient: map 88% reduce 0%
13/09/27 17:43:55 INFO mapred.JobClient: map 90% reduce 0%
13/09/27 17:43:56 INFO mapred.JobClient: map 91% reduce 0%
13/09/27 17:43:57 INFO mapred.JobClient: map 92% reduce 26%
13/09/27 17:43:58 INFO mapred.JobClient: map 95% reduce 26%
13/09/27 17:43:59 INFO mapred.JobClient: map 98% reduce 26%
13/09/27 17:44:00 INFO mapred.JobClient: map 100% reduce 26%
13/09/27 17:44:08 INFO mapred.JobClient: map 100% reduce 100%
13/09/27 17:44:11 INFO mapred.JobClient: Job complete: job_201309271742
13/09/27 17:44:11 INFO mapred.JobClient: Counters: 25
13/09/27 17:44:11 INFO mapred.JobClient: Job Counters
13/09/27 17:44:11 INFO mapred.JobClient:   Launched reduce tasks=1
13/09/27 17:44:11 INFO mapred.JobClient:   SLOTS_MILLIS_MAPS=88933
13/09/27 17:44:11 INFO mapred.JobClient:   Total time spent by all re
serving slots (ms)=0
13/09/27 17:44:11 INFO mapred.JobClient:   Total time spent by all ma
ving slots (ms)=0
13/09/27 17:44:11 INFO mapred.JobClient:   Launched map tasks=5
13/09/27 17:44:11 INFO mapred.JobClient:   Data-local map tasks=2
13/09/27 17:44:11 INFO mapred.JobClient:   SLOTS_MILLIS_REDUCES=22654
13/09/27 17:44:11 INFO mapred.JobClient: File Output Format Counters
13/09/27 17:44:11 INFO mapred.JobClient:   Bytes Written=47
13/09/27 17:44:11 INFO mapred.JobClient: FileSystemCounters
13/09/27 17:44:11 INFO mapred.JobClient:   FILE_BYTES_READ=2216
13/09/27 17:44:11 INFO mapred.JobClient:   HDFS_BYTES_READ=329139983
13/09/27 17:44:11 INFO mapred.JobClient:   FILE_BYTES_WRITTEN=130261
13/09/27 17:44:11 INFO mapred.JobClient:   HDFS_BYTES_WRITTEN=47
13/09/27 17:44:11 INFO mapred.JobClient: File Input Format Counters
13/09/27 17:44:11 INFO mapred.JobClient:   Bytes Read=329139468
13/09/27 17:44:11 INFO mapred.JobClient: Map-Reduce Framework
13/09/27 17:44:11 INFO mapred.JobClient:   Reduce input groups=2
13/09/27 17:44:11 INFO mapred.JobClient:   Map output materialized by
13/09/27 17:44:11 INFO mapred.JobClient:   Combine output records=71
13/09/27 17:44:11 INFO mapred.JobClient:   Map input records=9680132
13/09/27 17:44:11 INFO mapred.JobClient:   Reduce shuffle bytes=270
13/09/27 17:44:11 INFO mapred.JobClient:   Reduce output records=2
13/09/27 17:44:11 INFO mapred.JobClient:   Spilled Records=81
13/09/27 17:44:11 INFO mapred.JobClient:   Map output bytes=367843608
13/09/27 17:44:11 INFO mapred.JobClient:   Combine input records=9680
13/09/27 17:44:11 INFO mapred.JobClient:   Map output records=9680132
13/09/27 17:44:11 INFO mapred.JobClient:   SPLIT_RAW_BYTES=515
13/09/27 17:44:11 INFO mapred.JobClient:   Reduce input records=10

```

Figure 5. Hadoop Job Test

2.2 ExoGeni Testbed

When we started our development, we discussed the deployment problem with our Internet2 collaborators. As our application requires heavy use of VM resources, our Internet2 collaborators suggest us to use ExoGeni's resource to do the experiments across multiple sites.

After discussing with ExoGeni's manager, we found it impossible for us to use ExoGeni's hardware switch resources since they only support flat tag-based virtual Openflow-based network. This is not sufficient for us since we are going to build a hierarchical network topology.

Then we are suggested to use software switches instead of hardware switches to build the topology. This means that we run Open vSwitch instances inside VMs to forward the packets. Following this suggestion we have successfully built the topology we want, as shown in Figure 6. We sent test traffic through ping and everything worked fine.

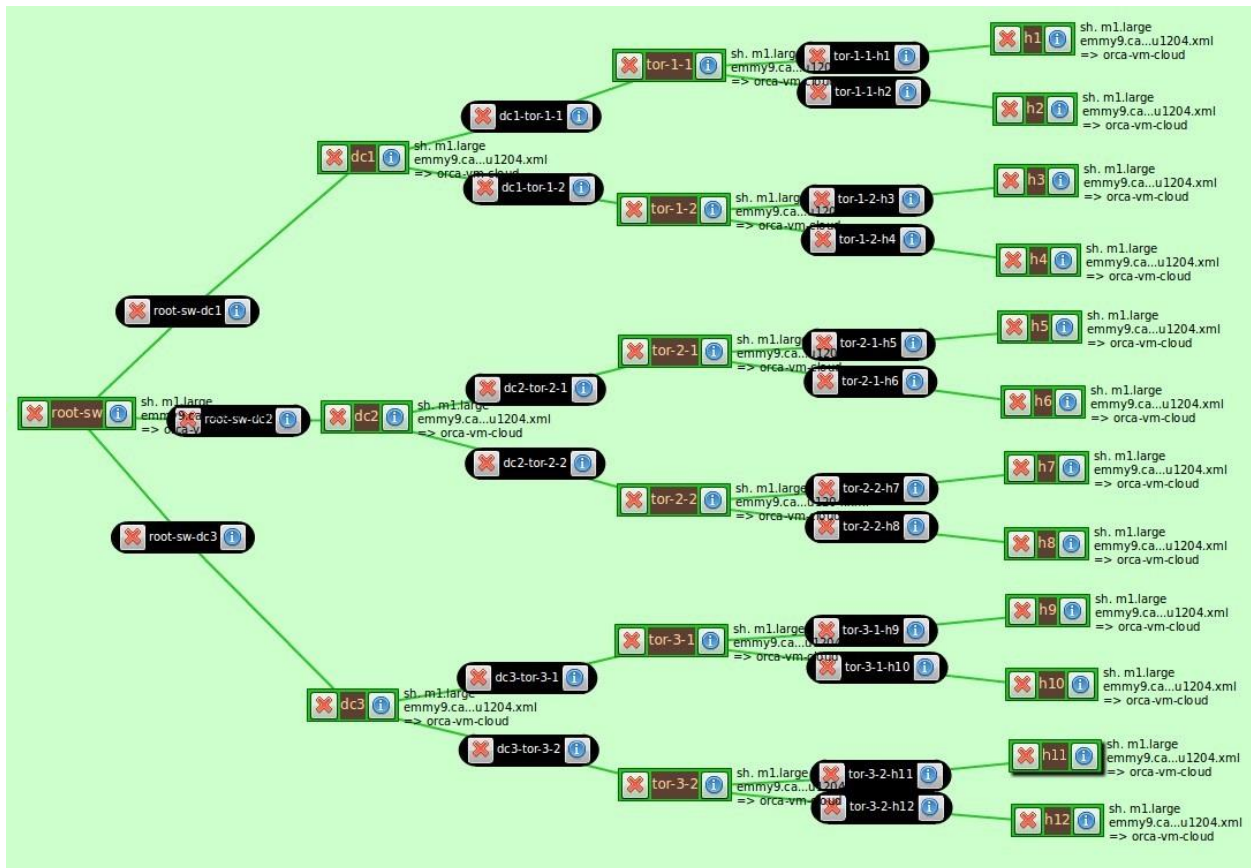


Figure 6. ExoGeni Testbed Topology

Unfortunately, when we were trying to connect these software switches to our controller, we found that the Floodlight controller cannot detect the links correctly. We believe it was not the problem of the Floodlight controller, but the testbed itself. We used wireshark and tcpdump toolkits to probe the problem. It seemed that our LLDP packet, which are used to construct the links by Floodlight, are absorbed by some unknown devices at somewhere inside the network. Since the underlying network is

hidden from us, we cannot further trace this problem. So we end up with using the local testbed to test our application and leave the wide-area development for the future work.

3. Other issues

Although we have passed all performed test suites, there are some potential improvements that could be done in the future. We have documented them in the code repository, under the /elf/doc directory.

We have revisited the original proposal because we could not get access to testbed resources that allow us to implement network bandwidth reservation. This issue has already been discussed during the follow-up meetings and the emails. Please see the design document for details.