

Dynamic Multipath Scheduling Protocol in SDN

Project Work of “Advanced Network
Architectures and Wireless Systems”

Nannini Alice
Poleggi Stefano

A.Y. 2020-2021



UNIVERSITÀ DI PISA



Used technologies

- VirtualBox 6.1
- Floodlight VM pre-configured with mininet, Open vSwitch, and Floodlight v1.1: for the tests, the VM has been executed with the maximum memory available (3562MB) and 4 processors. Two network adapters have been used: NAT and host-only adapter.
<https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/8650780/Floodlight+VM>
- Mininet 2.2.1
- Wireshark 2.6.6
- Iperf 2.0.5



Introduction

- DMSP is based on a multipath environment and makes real-time changes to select least congested links for routing data packets. The path selection and monitoring is done by a central SDN controller.
- We modified the java code of the project Floodlight to adapt it to the DMSP requirements.

S. A. Hussain, S. Akbar and I. Raza, "A dynamic multipath scheduling protocol (DMSP) for full performance isolation of links in software defined networking (SDN)," 2017 2nd Workshop on Recent Trends in Telecommunications Research (RTTR), Palmerston North, 2017, pp. 1-5, doi: 10.1109/RTTR.2017.7887866.



Topology

- We created the topology with redundant links for multipaths, using Mininet.
- We wrote a python code ("*fat-tree.py*") that implemented a "Fat Tree Topology" and we executed it from command line:

```
sudo mn --custom fat-tree.py  
--controller=remote,ip=192.168.56.109  
,port=6653 -topo=fat-tree
```



Building Multipaths

- We adapted the original module ***TopologyInstance*** in the package *net.floodlightcontroller.topology*: when calculating dijkstra shortest paths, we managed to store all the next-hops for each source-destination tuple that have minimum cost, instead of stopping at the first one found.
- We used the found next-hops to build all the possible paths and then we stored them in a cache.



Building Multipaths

The main functions in the code that we modified for calculating the multipaths are:

- *getRoute* [l.740-800]: get the possible multipaths from the cache to the *TopologyManager*
- *dijkstra* [l.495-560]: calculate all the minimum cost next-hops
- *buildroute* [l.644-714]: build all the possible paths from the next-hops of dijkstra

TopologyInstance

This is an extract of the code, in which we can see how the multiple next-hops are selected.

```
public class TopologyInstance {
    ...
    protected LoadingCache<RouteId, List<Route>> pathcache;
    ...
    protected BroadcastTree dijkstra(Cluster c, DatapathId root,
                                     Map<Link, Integer> linkCost,
                                     boolean isDstRooted) {
        HashMap<DatapathId, List<Link>> nexthoplinks = new HashMap<DatapathId, List<Link>>();
        ...
        if (ndist <= cost.get(neighbor)) {

            if( isLinkCongested(link,isDstRooted) )
                continue;

            cost.put(neighbor, ndist);
            List<Link> templinks = nexthoplinks.get(neighbor);
            if (templinks == null) templinks = new ArrayList<Link>();
            templinks.add(link);
            nexthoplinks.put(neighbor, templinks);
            NodeDist ndTemp = new NodeDist(neighbor, ndist);
            nodeeq.remove(ndTemp);
            nodeeq.add(ndTemp);
        }
        ....
    }
}
```

Scheduling of multipaths

- We have updated the ***TopologyManager*** module: after receiving the list of possible paths from the *TopologyInstance*, it schedules one for each source-destination host pair, using the Round Robin algorithm.
- The interested functions are:
 - class PathId [l.96-154]
 - getRoute [l.767-788]
 - scheduleNewRoute [l.791-807]

TopologyManager

```
public class TopologyManager implements IFloodlightModule, ITopologyService, IRoutingService, ILinkDiscoveryListener,
    IOFMessageListener {

    protected Map<RouteId,Integer> lastScheduledRoutes = new HashMap<RouteId,Integer>();
    protected Map<PathId,Integer> scheduledPaths = new HashMap<PathId,Integer>();
    ....
    @Override
    public Route getRoute(DatapathId src, OFPort srcPort, DatapathId dst, OFPort dstPort, U64 cookie,
        boolean tunnelEnabled) {
        TopologyInstance ti = getCurrentInstance(tunnelEnabled);
        List<Route> routes = ti.getRoute(null, src, srcPort, dst, dstPort, cookie);
        Route r = null;
        PathId pid = new PathId(src,srcPort,dst,dstPort);

        if (routes == null && src.equals(dst))
            r = ti.getWholeRoute(r, src, srcPort, dst, dstPort);
        ...
        r = routes.get(scheduledPaths.get(pid));
        ...
        return r;
    }

    private Route scheduleNewRoute(DatapathId src, OFPort srcPort, DatapathId dst, OFPort dstPort, List<Route> routes){
        PathId pid = new PathId(src,srcPort,dst,dstPort);
        RouteId rid = new RouteId(src,dst);
        ...
        int last = lastScheduledRoutes.get(rid);
        last = (last+1)%routes.size();
        lastScheduledRoutes.put(rid, last);
        scheduledPaths.put(pid,last);
        return routes.get(last);
    }
    ....
}
```

Statistics on bandwidth

- Since the DMSP protocol requires congestion control, so that different paths can be scheduled according to link available bandwidth, we included this part.
- We integrated the *statistics* module of a later version (present since 14 Dec 2015) of Floodlight to our system, which periodically collects statistics about link bandwidth.
- This service is exploited by the *TopologyInstance*: congested links do not contribute to path building.



Statistics on bandwidth

We added several classes in a new package `net.floodlightcontroller.statistics`:

- ***IStatisticsService***: interface of the service
- ***StatisticsCollector***: implements the interface and contains the threads that collect the bandwidth statistics
- ***SwitchPortBandwidth***: utility class to store statistics for each port
- The other classes in the package are for the Rest API service

Use of StatisticsCollector

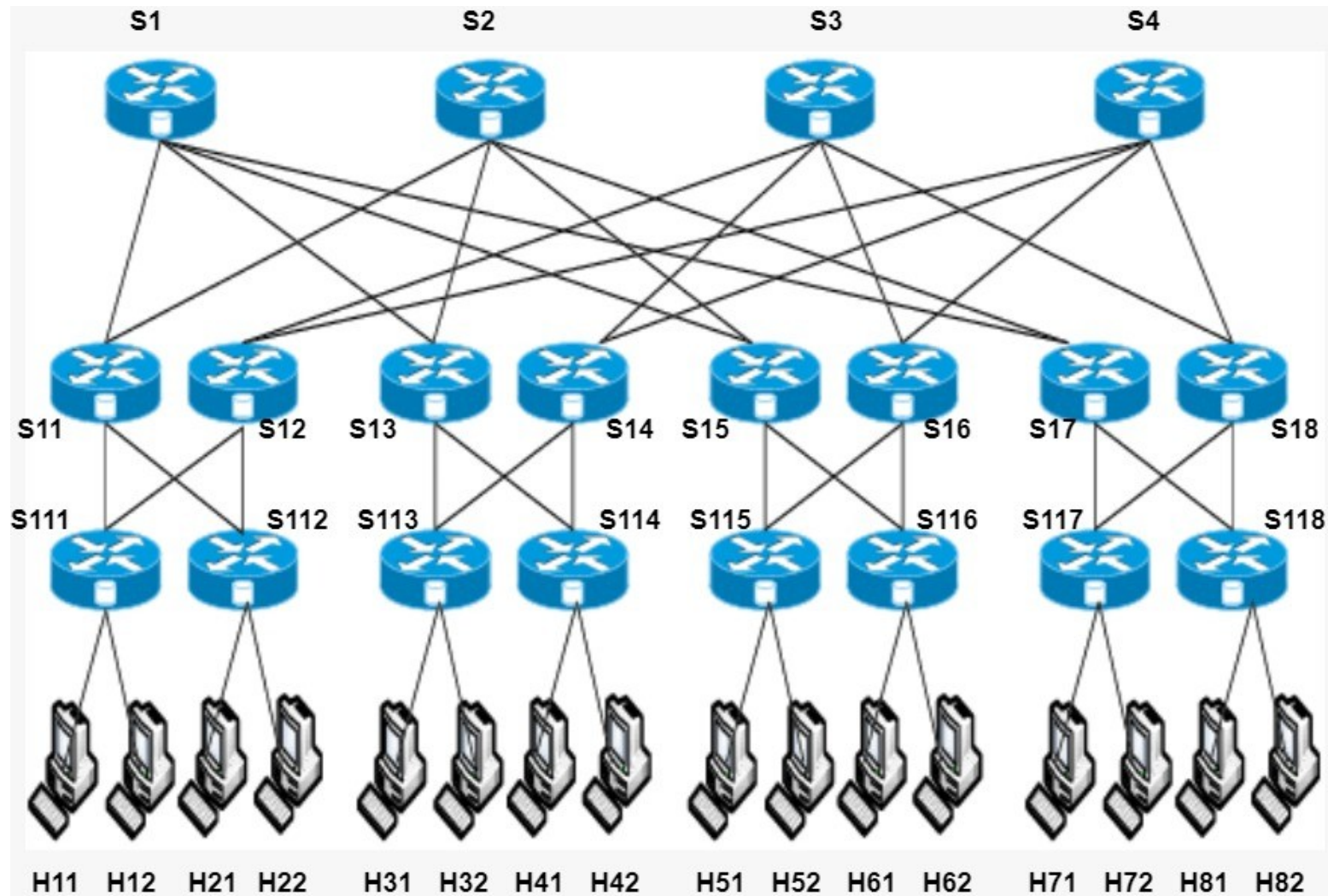
```
public class TopologyInstance {

    protected IStatisticsService statisticsCollectorService;
    ...

    protected BroadcastTree dijkstra(Cluster c, DatapathId root, Map<Link, Integer> linkCost, boolean isDstRooted) {
        ....
        if( isLinkCongested(link,isDstRooted) )
            continue;
        .....
    }

    protected boolean isLinkCongested(Link link, boolean isDstRooted){
        if(this.statisticsCollectorService != null){
            SwitchPortBandwidth spb = null;
            if(isDstRooted){
                spb = statisticsCollectorService.getBandwidthConsumption(link.getSrc(),link.getSrcPort());
            }else{
                spb = statisticsCollectorService.getBandwidthConsumption(link.getDst(),link.getDstPort());
            }
            // Check threshold;
            if (spb != null && spb.getBitsPerSecondTx().compareTo(statisticsCollectorService.getTxThreshold()) > 0){
                log.info(link+": TxBitsPerSec: "+spb.getBitsPerSecondTx()+"", over threshold;");
                return true;
            }
        }
        return false;
    }
}
```

Topology





Test on multipath

Two consecutive pings between hosts connected to the same switches go on different paths.

```
mininet> h11 ping -c 1 h41
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=20.7 ms

--- 10.0.0.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 20.792/20.792/20.792/0.000 ms
mininet> h11 ping -c 1 h42
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.
64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=27.0 ms

--- 10.0.0.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 27.043/27.043/27.043/0.000 ms
```

No.	Time	Source	Destination	Protocol	Length	Info	Interface id
5	-0.015173730	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s111-eth1
7	-0.003014273	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s111-eth4
9	-0.003007870	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s11-eth4
11	-0.000128085	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s11-eth1
13	-0.000123214	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s1-eth1
15	-0.000061989	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s1-eth2
17	-0.000057418	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s13-eth1
19	-0.000003400	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s13-eth3
1	0.000000000	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s114-eth4
3	0.000049787	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s114-eth1
4	0.000068082	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth1
2	0.005410383	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth4
20	0.005415825	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s13-eth3
18	0.005462957	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s13-eth1
16	0.005466405	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s1-eth2
14	0.005528212	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s1-eth1
12	0.005531807	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s11-eth1
10	0.005562314	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s11-eth4
8	0.005564621	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth4
6	0.005591334	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth1
28	9.025874915	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request ...	s111-eth1
30	9.031446361	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request ...	s111-eth3
21	9.031455877	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request ...	s12-eth4
24	9.035961265	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request ...	s12-eth2
32	9.035966330	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request ...	s4-eth1
34	9.036013873	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request ...	s4-eth2
22	9.036016628	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request ...	s14-eth2
26	9.036057355	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request ...	s14-eth3
36	9.036060245	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request ...	s114-eth3
38	9.036088265	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request ...	s114-eth2
39	9.036103994	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth2
37	9.037833543	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth3
27	9.037837627	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply ...	s14-eth3
23	9.037878038	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply ...	s14-eth2
35	9.037880917	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply ...	s4-eth2
33	9.042136762	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply ...	s4-eth1
25	9.042143530	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply ...	s12-eth2
40	9.052784038	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply ...	s12-eth4
31	9.052789344	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth3
29	9.052887863	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth1



Test on congestion

Two consecutive pings between the same hosts go on the same path.

```
mininet> h11 ping -c 1 h41
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=22.3 ms

--- 10.0.0.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 22.318/22.318/22.318/0.000 ms
mininet> h11 ping -c 1 h41
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=10.6 ms

--- 10.0.0.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 10.695/10.695/10.695/0.000 ms
mininet>
```

No.	Time	Source	Destination	Protocol	Length	Info	Interface id
1	0.000000000	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s111-eth1
3	0.009411146	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s111-eth4
5	0.009416482	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s11-eth4
11	0.012970510	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s11-eth1
17	0.012973333	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s1-eth1
9	0.013048574	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s1-eth2
15	0.013051999	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s13-eth1
19	0.013104538	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s13-eth3
7	0.013107947	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s114-eth4
13	0.013142874	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s114-eth1
14	0.013164574	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth1
8	0.022139925	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth4
20	0.022145679	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s13-eth3
16	0.022192290	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s13-eth1
10	0.022201353	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s1-eth2
18	0.022233785	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s1-eth1
12	0.022236415	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s11-eth1
6	0.022262533	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s11-eth4
4	0.022265124	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth4
2	0.022294743	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth1
25	3.732317601	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s111-eth1
27	3.740094534	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s111-eth4
29	3.740099209	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s11-eth4
31	3.740149703	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s11-eth1
21	3.740152736	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s1-eth1
23	3.740187009	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s1-eth2
33	3.740189689	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s13-eth1
35	3.740216859	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s13-eth3
37	3.740219551	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s114-eth4
39	3.740244542	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s114-eth1
40	3.740259471	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth1
38	3.742834521	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth4
36	3.742840013	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s13-eth3
34	3.742892054	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s13-eth1
24	3.742895173	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s1-eth2
22	3.742926495	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s1-eth1
32	3.742928988	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s11-eth1
30	3.742955196	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s11-eth4
28	3.742957595	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth4
26	3.742990196	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth1

Test on congestion

```

"Node: h11"
root@floodlight:~/project/floodlight_project# iperf -c 10.0.0.7 -u -b 1000M
Client connecting to 10.0.0.7, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 208 KByte (default)

[ 79] local 10.0.0.1 port 36787 connected with 10.0.0.7 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 79] 0.0-10.0 sec   863 MBytes  724 Mbits/sec
[ 79] Sent 615326 datagrams
[ 79] Server Report:
[ 79] 0.0-10.0 sec   834 MBytes  701 Mbits/sec    0.000 ms 20638/615325 (3.4%)
[ 79] 0.0-10.0 sec  127 datagrams received out-of-order
root@floodlight:~/project/floodlight_project#

```

```

"Node: h41"
root@floodlight:~/project/floodlight_project# iperf -s -u
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)

[ 79] local 10.0.0.7 port 5001 connected with 10.0.0.1 port 36787
[ ID] Interval      Transfer    Bandwidth      Jitter    Lost/Total Datagrams
[ 79] 0.0-10.0 sec   834 MBytes  701 Mbits/sec   0.001 ms 20638/615325 (3.4%)
[ 79] 0.0-10.0 sec   127 datagrams received out-of-order

```

We congest the path between h11 and h41 with the command *iperf*.

The controller will detect the congestion on the links and compute a different path for the hosts communication.

```

mininet> h11 ping -c 1 h41
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=13.1 ms

--- 10.0.0.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 13.141/13.141/13.141/0.000 ms
mininet> h11 ping -c 1 h41
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=14.0 ms

--- 10.0.0.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 14.014/14.014/14.014/0.000 ms
mininet>

```


Test on congestion

No.	Time	Source	Destination	Protocol	Length	Info	Interface id
2	-0.067705264	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s111-eth1
6	-0.002210492	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s111-eth4
16	-0.002203960	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s11-eth4
18	-0.000806475	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s11-eth1
14	-0.000800061	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s1-eth1
8	-0.000009296	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s1-eth2
1	0.000000000	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s13-eth1
4	0.000041933	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s13-eth3
12	0.000047434	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s114-eth4
10	0.000083282	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s114-eth1
11	0.000111036	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth1
13	0.003600459	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth4
5	0.003609796	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s13-eth3
20	0.095849187	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s13-eth1
9	0.095855913	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s1-eth2
15	0.096130884	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s1-eth1
19	0.096139848	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s11-eth1
17	0.096198366	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s11-eth4
7	0.096203820	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth4
3	0.096268148	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth1
21	179.2681528...	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s111-eth1
22	179.2725070...	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s111-eth4
31	179.2725119...	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s11-eth4
25	179.2770466...	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s11-eth2
38	179.2770520...	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s2-eth1
37	179.2771008...	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s2-eth2
33	179.2771037...	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s13-eth2
32	179.2771716...	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s13-eth3
27	179.2771746...	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s114-eth4
23	179.2772022...	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request ...	s114-eth1
24	179.2772168...	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth1
28	179.2780081...	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s114-eth3
34	179.2780120...	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s14-eth3
35	179.2803867...	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s14-eth1
39	179.2803915...	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s3-eth2
40	179.2808016...	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s3-eth1
30	179.2808054...	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s12-eth1
29	179.2812245...	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s12-eth4
26	179.2812280...	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth3
36	179.2812689...	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply ...	s111-eth1

Before the congestion

After the congestion



Parameters

To study better the network we could modify some parameters:

- Add the option *–link tc,bw=10* to the mininet command, to limit the bandwidth of the links
- Change consequently the **TX_THRESHOLD** of the interface *IstatisticsService*
- Change the **FLOWMOD_DEFAULT_IDLE_TIMEOUT** in the *ForwardingBase* class, to alterate the timeout of the flow table entries
- Change the *expireAfterWrite* timeout of the *PathCache* in the *TopologyInstance* class.