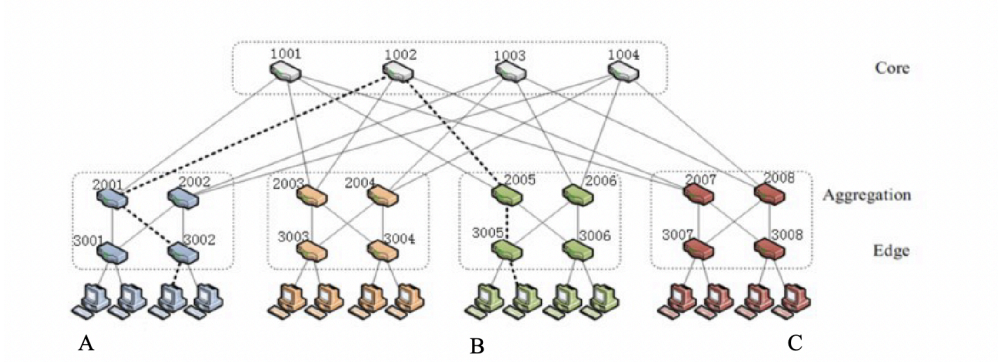
ECE 158B Project #2

Part 1:

Topology



References: Host A: h1, Host B: h9, Host C: h16

Code (Bandwidth and Latency already configured)

Users > praveenswaminathan > Downloads > FatTreeTopology.py > ...

```python
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import OVSSwitch, OVSController, OVSBridge
from mininet.link import TCLink
from mininet.cli import CLI

class FatTree(Topo):
    def build(self):
        c = []
        a = []
        e = []
        h = []

        clen = 4
        alen = 8
        elen = 8
        hlen = 16

        # Enable STP on all switches
        switchOpts = {'cls':OVSBridge, 'stp':1}

        for i in range(clen):
            coresw = self.addSwitch('c{}'.format(i+1), **switchOpts)
            c.append(coresw)

        for i in range(alen):
            aggsw = self.addSwitch('a{}'.format(i+1), **switchOpts)
            a.append(aggsw)

        for i in range(elen):
            edgesw = self.addSwitch('e{}'.format(i+1), **switchOpts)
            e.append(edgesw)

        for i in range(hlen):
            host = self.addHost('h{}'.format(i+1))
            h.append(host)

        #Core Switches
        for i in range(2):
            self.addLink(c[i], a[0], bw=100, delay='1ms')
            self.addLink(c[i], a[2], bw=100, delay='1ms')
            self.addLink(c[i], a[4], bw=100, delay='1ms')
            self.addLink(c[i], a[6], bw=100, delay='1ms')
        for i in range(2, 4):
            self.addLink(c[i], a[1], bw=100, delay='1ms')
            self.addLink(c[i], a[3], bw=100, delay='1ms')
            self.addLink(c[i], a[5], bw=100, delay='1ms')
            self.addLink(c[i], a[7], bw=100, delay='1ms')
```

```
49
50          #Aggregation switch connections
51          for i in range(0, 8, 2):
52              self.addLink(a[i], e[i], bw=100, delay='1ms')
53              self.addLink(a[i], e[i+1], bw=100, delay='1ms')
54              self.addLink(a[i+1], e[i], bw=100, delay='1ms')
55              self.addLink(a[i+1], e[i+1], bw=100, delay='1ms')
56
57          #Edge switch connections
58          for i in range(8):
59              self.addLink(e[i], h[2*i], bw=100, delay='1ms')
60              self.addLink(e[i], h[2*i+1], bw=100, delay='1ms')
61
62
63
64  topos = {'mytopo':(lambda:FatTree())}
65
66  if __name__ == '__main__':
67      topo = FatTree()
68      net = Mininet(topo=topo, controller=OVSController, link=TCLink)
69      net.start()
70
71      CLI(net)
72      net.stop()
73
```

Network Connectivity Check

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13 h14 h15 h16
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13 h14 h15 h16
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12 h13 h14 h15 h16
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12 h13 h14 h15 h16
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12 h13 h14 h15 h16
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h13 h14 h15 h16
h13 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h14 h15 h16
h14 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h15 h16
h15 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h16
h16 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Results: 0% dropped (240/240 received)
mininet>
```

Part 2:

Mice (pings) and Elephant (iperf) flows separately

```
mininet> h1 ping -c 100 h9
64 bytes from 10.0.0.9: icmp_seq=80 ttl=64 time=12.3 ms
64 bytes from 10.0.0.9: icmp_seq=81 ttl=64 time=12.5 ms
64 bytes from 10.0.0.9: icmp_seq=82 ttl=64 time=12.3 ms
64 bytes from 10.0.0.9: icmp_seq=83 ttl=64 time=12.2 ms
64 bytes from 10.0.0.9: icmp_seq=84 ttl=64 time=12.2 ms
64 bytes from 10.0.0.9: icmp_seq=85 ttl=64 time=12.3 ms
64 bytes from 10.0.0.9: icmp_seq=86 ttl=64 time=12.3 ms
64 bytes from 10.0.0.9: icmp_seq=87 ttl=64 time=12.3 ms
64 bytes from 10.0.0.9: icmp_seq=88 ttl=64 time=12.2 ms
64 bytes from 10.0.0.9: icmp_seq=89 ttl=64 time=12.3 ms
64 bytes from 10.0.0.9: icmp_seq=90 ttl=64 time=12.3 ms
64 bytes from 10.0.0.9: icmp_seq=91 ttl=64 time=12.3 ms
64 bytes from 10.0.0.9: icmp_seq=92 ttl=64 time=12.4 ms
64 bytes from 10.0.0.9: icmp_seq=93 ttl=64 time=12.3 ms
64 bytes from 10.0.0.9: icmp_seq=94 ttl=64 time=12.4 ms
64 bytes from 10.0.0.9: icmp_seq=95 ttl=64 time=12.2 ms
64 bytes from 10.0.0.9: icmp_seq=96 ttl=64 time=12.3 ms
64 bytes from 10.0.0.9: icmp_seq=97 ttl=64 time=12.3 ms
64 bytes from 10.0.0.9: icmp_seq=98 ttl=64 time=12.4 ms
64 bytes from 10.0.0.9: icmp_seq=99 ttl=64 time=12.2 ms
64 bytes from 10.0.0.9: icmp_seq=100 ttl=64 time=12.3 ms

--- 10.0.0.9 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99217ms
rtt min/avg/max/mdev = 12.223/12.316/13.895/0.192 ms
mininet>
```

```
mininet> h9 iperf -s &
mininet> h1 iperf -c h9 -n 100M
------------------------------------------------------------
Client connecting to 10.0.0.9, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.1 port 37382 connected with 10.0.0.9 port 5001 (icwnd/mss/irtt=14/1448
/14946)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-8.8740 sec   100 MBytes  94.5 Mbits/sec
mininet>
```

Mice and Elephant Flows Together

```
mininet> h1 sh -c 'ping -c 100 h9 & iperf -c h9 -n 100M & wait'
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
------------------------------------------------------------
Client connecting to 10.0.0.9, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
64 bytes from 10.0.0.9: icmp_seq=1 ttl=64 time=23.6 ms
[  1] local 10.0.0.1 port 44990 connected with 10.0.0.9 port 5001 (icwnd/mss/irtt=14/1448
/22902)
64 bytes from 10.0.0.9: icmp_seq=2 ttl=64 time=102 ms
64 bytes from 10.0.0.9: icmp_seq=3 ttl=64 time=151 ms
64 bytes from 10.0.0.9: icmp_seq=4 ttl=64 time=204 ms
64 bytes from 10.0.0.9: icmp_seq=5 ttl=64 time=251 ms
64 bytes from 10.0.0.9: icmp_seq=6 ttl=64 time=301 ms
64 bytes from 10.0.0.9: icmp_seq=7 ttl=64 time=352 ms
64 bytes from 10.0.0.9: icmp_seq=8 ttl=64 time=402 ms
64 bytes from 10.0.0.9: icmp_seq=9 ttl=64 time=452 ms
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-8.9228 sec   100 MBytes  94.0 Mbits/sec
64 bytes from 10.0.0.9: icmp_seq=10 ttl=64 time=20.4 ms
64 bytes from 10.0.0.9: icmp_seq=11 ttl=64 time=20.4 ms
64 bytes from 10.0.0.9: icmp_seq=12 ttl=64 time=20.4 ms
64 bytes from 10.0.0.9: icmp_seq=13 ttl=64 time=20.4 ms
64 bytes from 10.0.0.9: icmp_seq=14 ttl=64 time=20.5 ms
```

```
--- 10.0.0.9 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99247ms
rtt min/avg/max/mdev = 20.357/41.037/451.706/76.713 ms
mininet>
```

The difference between this setup, with simultaneous mice and elephant traffic, as compared to separate flows, is evident in the time that it takes for the first 10 packets in the ping. As both flows are competing with each other, neither flow is optimized. Because elephant flows need high throughput, the latency is longer, resulting in delays of the ping that are around 200-400ms, which is extremely slow. Similarly, but quite less in the impact as compared to the ping traffic, is the lower bandwidth for the elephant flow, going from 94.5Mbps to 94Mbps, which is a small, but still noticeable difference. Furthermore, even after the download is complete, the time per ping is still much slower than if it was alone, going from ~12ms per ping in the single flow, to ~20 ms per ping with simultaneous flows. These discrepancies can be attributed to TCP's inability to accommodate for both flows' requirements (high throughput vs low latency), which is why DC-TCP is utilized in most data center topologies.