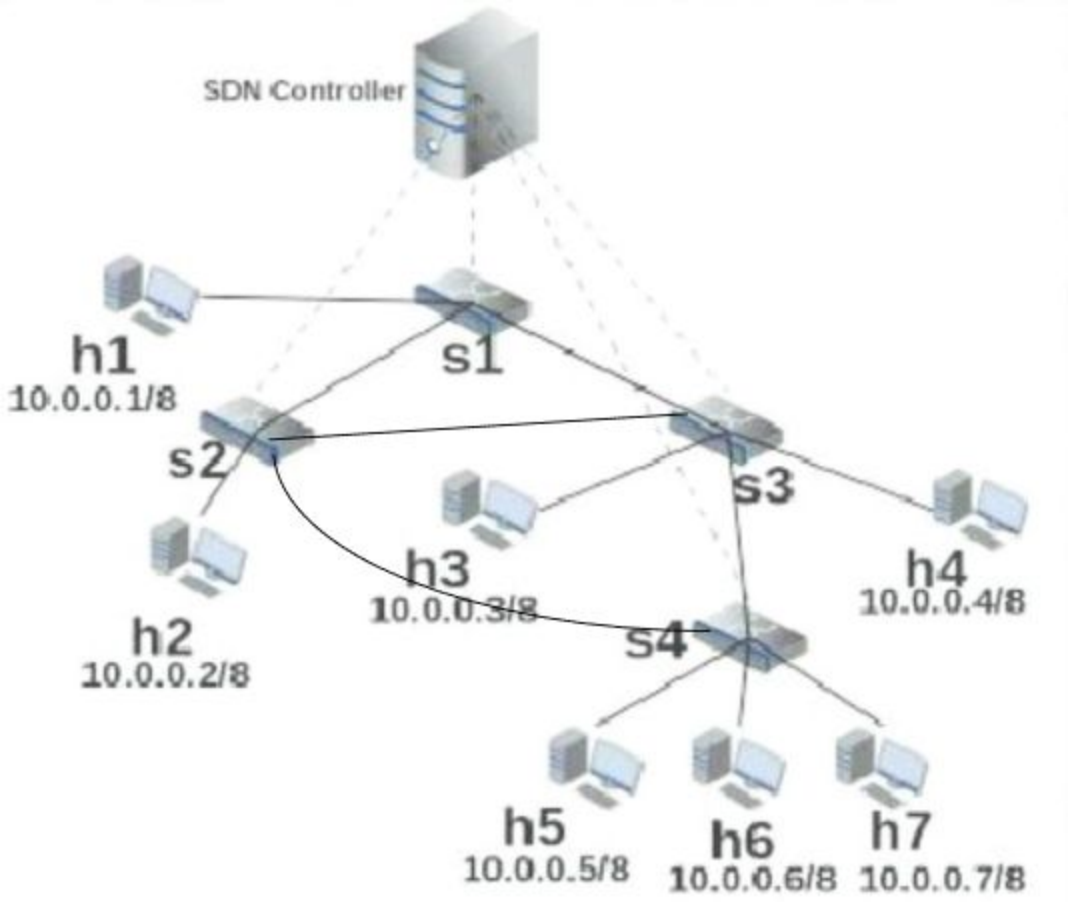
پروژه شماره 4 شبکه های کامپیوتری

(قبل از اصلاحیه)

محسن فیاض 810196650 ژیوار صورتسی 810196502

SDN - Ryu



مراحل پیاده‌سازی

❖ نصب

در ابتدا با دستورات زیر نصب را شروع کردیم.

```
Cd
pip install eventlet msgpack-python netaddr oslo.config routes six
webobgit
clone git://github.com/osrg/ryu.gitcd
ryupython ./setup.py install
```

سپس دستورات تست را خواستیم اجرا کنیم که به مشکلات متعدد خوردیم و شروع به نصب نیازمندی هایی که ذکر نشده بود کردیم.

```
ERROR: ryu 4.34 requires
eventlet!=0.18.3,!=0.20.1,!=0.21.0,!=0.23.0,>=0.18.2, which is not
installed.
ERROR: ryu 4.34 requires msgpack<1.0.0,>=0.3.0, which is not installed.
ERROR: ryu 4.34 requires netaddr, which is not installed.
ERROR: ryu 4.34 requires oslo.config>=2.5.0, which is not installed.
ERROR: ryu 4.34 requires ovs>=2.6.0, which is not installed.
ERROR: ryu 4.34 requires routes, which is not installed.
ERROR: ryu 4.34 requires tinyrpc==0.9.4, which is not installed.
```

سپس برای نصب mininet از گیت فایل ها را گرفتیم و با کد های زیر نصبش کردیم.

```
util/install.sh -fnv
```

❖ طراحی توپولوژی

```
class Topology(Topo):
   def __init__(self):
       Topo.__init__(self)
       # hosts
       h1 = self.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
       h2 = self.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
       h3 = self.addHost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
       h4 = self.addHost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)
       h5 = self.addHost('h5', cls=Host, ip='10.0.0.5', defaultRoute=None)
       h6 = self.addHost('h6', cls=Host, ip='10.0.0.6', defaultRoute=None)
       h7 = self.addHost('h7', cls=Host, ip='10.0.0.7', defaultRoute=None)
       # switches
       s1 = self.addSwitch('s1')
       s2 = self.addSwitch('s2')
       s3 = self.addSwitch('s3')
       s4 = self.addSwitch('s4')
       # links
       self.addLink(h1, s1, cls=TCLink, bw=1, delay='1ms', loss=0)
       self.addLink(s2, s1, cls=TCLink, bw=1, delay='1ms', loss=0)
       self.addLink(h2, s2, cls=TCLink, bw=1, delay='1ms', loss=0)
       # self.addLink(s2, s3, cls=TCLink, bw=1, delay='1ms', loss=0)
       self.addLink(s1, s3, cls=TCLink, bw=1, delay='1ms', loss=0)
       # self.addLink(s2, s4, cls=TCLink, bw=1, delay='1ms', loss=0)
       self.addLink(h3, s3, cls=TCLink, bw=1, delay='1ms', loss=0)
       self.addLink(h4, s3, cls=TCLink, bw=1, delay='1ms', loss=0)
       self.addLink(s3, s4, cls=TCLink, bw=1, delay='1ms', loss=0)
       self.addLink(h5, s4, cls=TCLink, bw=1, delay='1ms', loss=0)
       self.addLink(h6, s4, cls=TCLink, bw=1, delay='1ms', loss=0)
       self.addLink(h7, s4, cls=TCLink, bw=1, delay='1ms', loss=0)
```

❖ تنظیم رندوم bandwidth

```
def change_bw_timer_task(net):
    global running
    while running:
        for link in net.links:
            bw = random.randint(MIN_BW, MAX_BW)
            link.intf1.params["bw"] = bw
        print("net.links[0].intf1: ", net.links[0].intf1.params["bw"])
        time.sleep(BW_INTERVAL)
```

بعد از جستجوی فراوان در کدهای mininet فهمیدیم که bw در

```
Link.intf1.params["bw"]
```

ذخیره می شود. بنابراین با استفاده از thread هر 10 ثانیه این مقدار را به صورت رندوم برای تمام لینک ها مشخص کردیم.

❖ ارسال بسته TCP

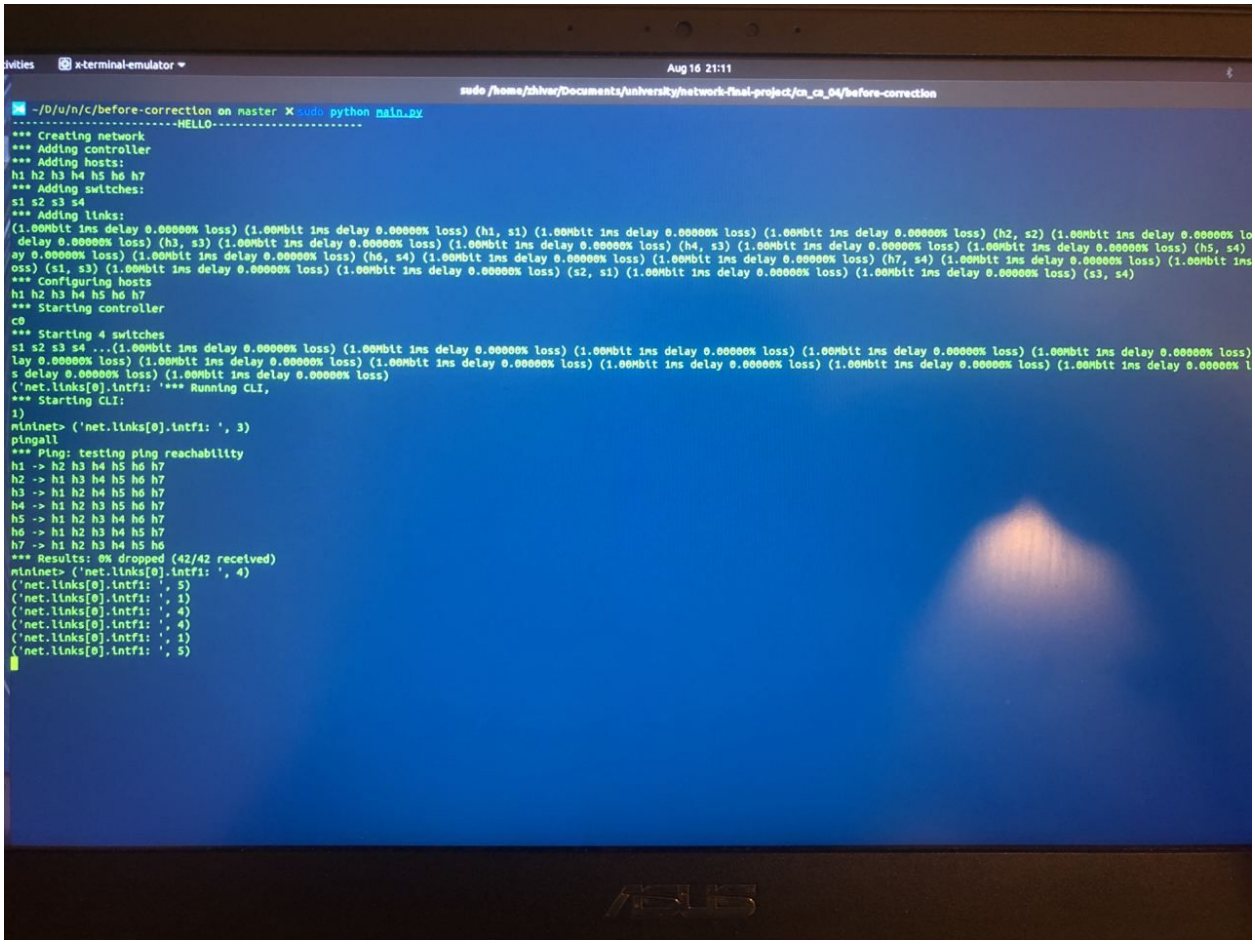
```
def send_data_timer_task(net):
   global running
   while running:
       save = None
       print("--sending TCP messages")
       for host_index in range(1, 8):
           h = net.get('h'+str(host_index))

           target_host_index = random.randint(1, 7)
           while target_host_index == host_index:
               target_host_index = random.randint(1, 7)

           target_h = net.get("h"+str(target_host_index))
           save = target_h.IP()
           r = h.cmd('iperf -c %s -n 100000' % target_h.IP())
           print(r)
       print("--sent TCP messages: ", save)
       time.sleep(SEND_TCP_INTERVAL)
```

باز هم بعد از جستجو های فراوان راهی که برای ارسال 100 کیلوبایت اطلاعات یافتیم این بود که ابتدا برای هر host یک cmd باز کرده و داخلش با iperf بسته ای را به ip یک host دیگر بفرستیم.

نتایج



Tehran, Iran