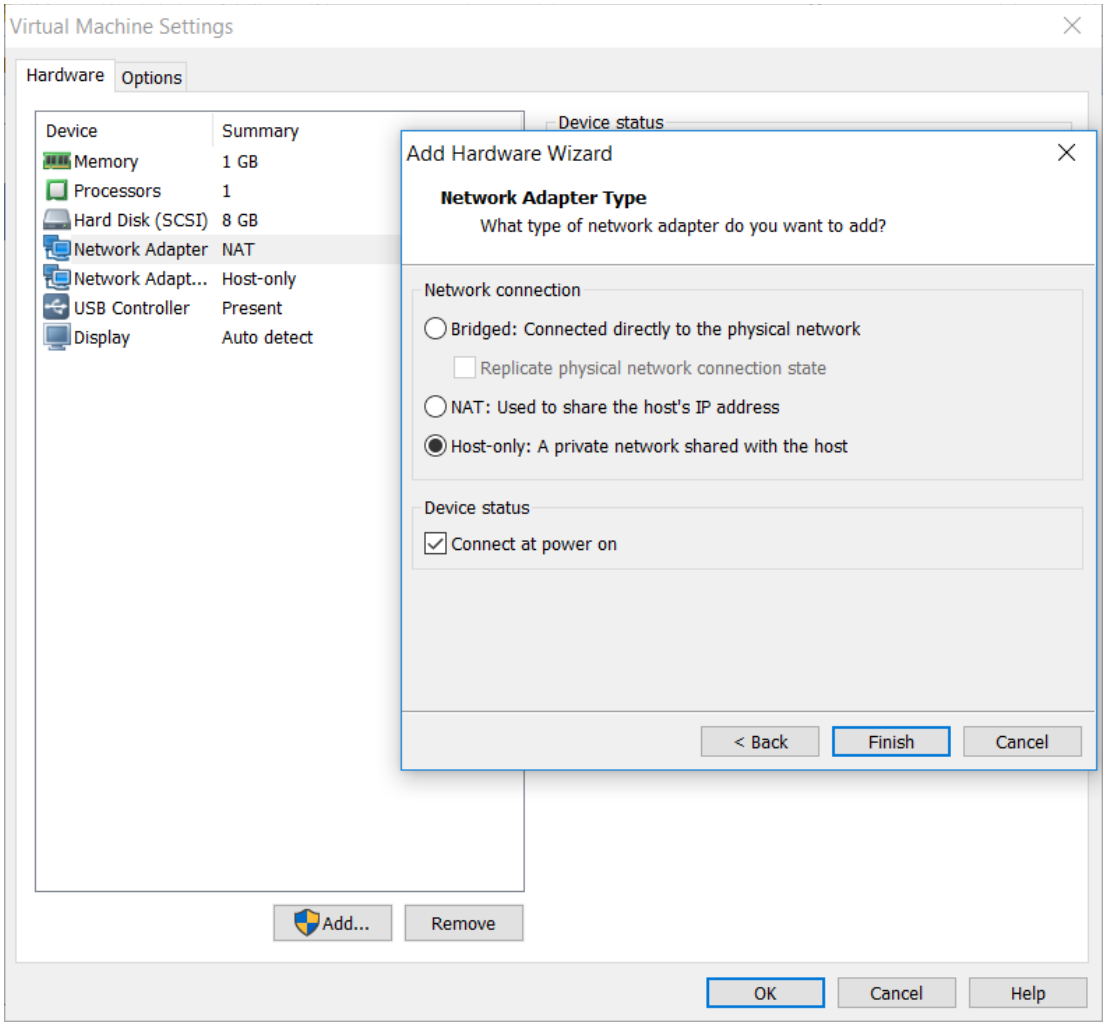
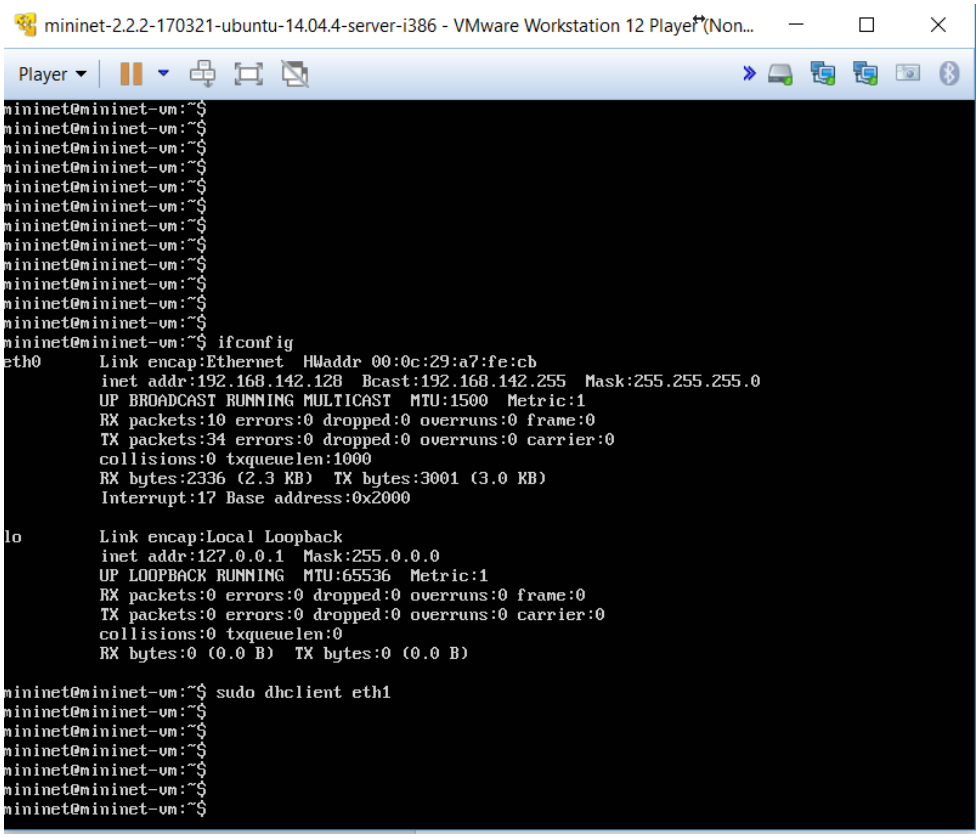
EMULACIJA SDN MREŽA U MININETU

PRIPREMA ZA VJEŽBU

1. Preuzeti Mininet virtuelnu mašinu sa sajta: https://github.com/mininet/mininet/wiki/Mininet-VM-Images
2. Importovati Mininet VM u VMWare.
3. Dodati još jedan mrežni interfejs virtuelnoj mašini koji bi trebao da bude "Host-only" tipa (**Edit virtual machine settings -> Add -> Network Adapter**).



4. Pokrenuti VM. Unijeti komnandu: `sudo dhclient eth1`



5. Na host mašini pokrenuti MobeXterm i unijeti komandu:

   `ssh -X mininet@adresa_eth0_interfejsa`

6. Unijeti login password: **mininet**
7. Instalirati grafički editor **gedit** sa:

   ```
   sudo apt-get update

   sudo apt-get install gedit
   ```

OSNOVNE MININET KOMANDE

Da bi se kreirala najjednostavnija mrežna topologija u Mininetu potrebno je unijeti komandu:

```
$ sudo mn
```

Kreirana topologija sadrži jedan OpenFlow kernel *switch*, dva hosta i referentnu implementaciju OpenFlow kontrolera. Unosom `help` komande u komandnoj liniji moguće je provjeriti set dostupnih Mininet komandi. U nastavku je dat pregled nekih od njih.

Prikaz seta mrežnih čvorova:

```
mininet> nodes
```

Za dobijanje detaljnijih infrmacija o mrežnim čvorovima koristiti komandu *dump.*

Prikaz topologije:

```
mininet> net
```

Jednostavne komande unutar čvora:

Ukoliko je prvi string ukucan u Mininet komandnoj liniji ime hosta, *switch-a* ili kontrolera , komanda u nastavku izvršiće se na tom navedenom čvoru. Na primjer, za prikaz informacija o mrežnim interfejsima hosta *h1* dovoljno je ukucati:

```
mininet> h1 ifconfig –a
```

Ukoliko se u prethodnoj komandi *h1* zamjeni sa *s1* , rezultat će biti prikaz detalja o interfejsima *switch*-a. Slično, listu procesa koji se izvršavaju na hostu moguće je dobiti unosom: *ps –a* .

Zadatak 1: Odrediti IP i MAC adrese mrežnih uređaja. Na koje portove *switch*-a su povezani hostovi i kontroler?

Testiranje konektivnosti između hostova

Sledećom komandom moguće je izvršiti pingovanje hosta 2 sa hosta 1.

```
mininet> h1 ping –c5 h2
```

Zadatak 2: Izanalizirati kašnjenje za svaki od 5 poslatih paketa. Koliko je iznosio RTT za svaki od paketa?

Zadatak 3: Trebalo bi da primjetite da je RTT veći za prvi paket nego sa ostala 4 paketa. Kako objašnjavate ovo zapažanje?

Jednostavan način da testirate konektivnost između svih čvorova u Mininetu je korišćenje *pingall* komande, koja pokreće ping process između svakog para hostova u mreži.

XTerm terminal

Za prikaz xterm terminala za host 1 dovoljno je unijeti komandu:

```
mininet> xterm h1
```

Python interpreter

Ukoliko je prvi dio komande string *py* , Mininet interpretira unijeti tekst kao Python komandu. Ovo značajno olakšava različite oblike manipulacija mrežnim uređajima, kao što su hostovi, *switch*-evi i kontroleri. Na primjer, informacija o IP adresi hosta *h1* može se dobiti sa:

```
mininet> py h1.IP()
```

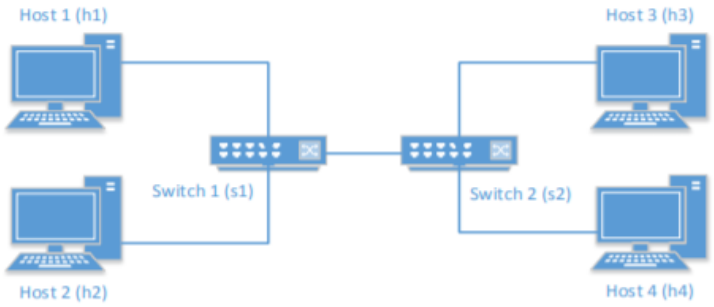
Stopiranje emulatora i oslobađanje resursa

Izlazak iz Mininet komandnog okruženja ostvaruje se komandom `exit`. Strogo se preporučuje naknadno oslobađanje resursa koji su korišćeni tokom emulacije, što se postiže komandom `sudo mn -c`.

KREIRANJE PROIZVOLJNIH TOPOLOGIJA U MININETU

Mininet podržava jednostavan Python API koji omogućava kreiranje proizvoljnih mrežnih topologija sa svega nekoliko linija koda. Na primjer, Slika 1 prikazuje topologiju sa dva *switch*-a i četiri hosta. Odgovarajući kod (Slika 2) sastoji se od klase MyFirstTopo koja nasleđje klasu Topo, definisanu u Mininet bibliotekama. U nastavku vježbe poželjno je koristiti SSH konekciju prema Mininet virtuelnoj mašini kako bi se mogli koristiti grafički tekstualni editori poput *gedit*. Unesite kod u jedan fajl i sačuvajte ga pod nazivom *myfirsttopo.py* u RM direktorijumu. Podesite radni direktorijum na RM direktorijum, a zatim unesite komandu:

```
mininet> sudo mn --custom myfirsttopo.py --topo myfirsttopo
```



Slika 1. Primjer jednostavne mrežne topologije.

```
1.  from mininet.topo import Topo
2.
3.  class MyFirstTopo( Topo ):
4.      "Simple topology example."
5.      def __init__( self ):
6.          "Create custom topo."
7.          # Initialize topology
8.          Topo.__init__( self )
9.          # Add hosts and switches
10.         h1 = self.addHost( 'h1' )
11.         h2 = self.addHost( 'h2' )
12.         h3 = self.addHost( 'h3' )
13.         h4 = self.addHost( 'h4' )
14.         leftSwitch = self.addSwitch( 's1' )
15.         rightSwitch = self.addSwitch( 's2' )
16.         # Add links
17.         self.addLink( h1, leftSwitch )
18.         self.addLink( h2, leftSwitch )
19.         self.addLink( leftSwitch, rightSwitch )
20.         self.addLink( rightSwitch, h3 )
21.         self.addLink( rightSwitch, h4 )
22.
23. topos = { 'myfirsttopo': ( lambda: MyFirstTopo() ) }
```

Slika 2: Python kod za kreiranje topologije sa Slike 1.

Zadatak 4: Na Slici 1 naznačiti IP i MAC adrese hostova. Takođe naznačiti nazive i MAC adrese interfejsa *switch*-eva na koje su povezani hostovi.

Za brzo testiranje ove topologije dovoljno je koristiti opciju `--test` u Mininetu:

```
mininet> sudo mn --custom myfirsttopo.py --topo myfirsttopo --test pingall
```

Sada pretpostavimo da želimo da ostvarimo sledeće:

1. Kreiramo topologiju sa Slike 1.
2. Pokrenemo mrežu.
3. Dobijemo detaljne informacije o konektivnosti između svih uređaja u mreži.
4. Testiramo konektivnost između hostova.

Automatizovaćemo navedene radnje sa Python skriptom prikazanom na Slici 3. Skriptu ćemo kreirati nadogradnjom koda sa Slike 2, i sačuvati pod nazivom *myfirstexpr.py*. Pokretanje skripte se vrši na sledeći način:

```
mininet> sudo python myfirstexpr.py
```

Nakon eksperimenta izbrišite trenutnu emulaciju sa komandom *mn -c*.

```
1.  #!/usr/bin/python
2.
3.  from mininet.topo import Topo
4.  from mininet.net import Mininet
5.  from mininet.util import dumpNodeConnections
6.  from mininet.log import setLogLevel
7.
8.  class MyFirstTopo( Topo ):
9.      "Simple topology example."
10.     def __init__( self ):
11.         "Create custom topo."
12.         # Initialize topology
13.         Topo.__init__( self )
14.         # Add hosts and switches
15.         h1 = self.addHost( 'h1' )
16.         h2 = self.addHost( 'h2' )
17.         h3 = self.addHost( 'h3' )
18.         h4 = self.addHost( 'h4' )
19.         leftSwitch = self.addSwitch( 's1' )
20.         rightSwitch = self.addSwitch( 's2' )
21.         # Add links
22.         self.addLink( h1, leftSwitch )
23.         self.addLink( h2, leftSwitch )
24.         self.addLink( leftSwitch, rightSwitch )
25.         self.addLink( rightSwitch, h3 )
26.         self.addLink( rightSwitch, h4 )
27.
28. def runExperiment():
29.     "Create and test a simple experiment"
30.     topo = MyFirstTopo( )
31.     net = Mininet(topo)
32.     net.start()
33.     print "Dumping host connections"
34.     dumpNodeConnections(net.hosts)
35.     print "Testing network connectivity"
36.     net.pingAll()
37.     net.stop()
38.
39. if __name__ == '__main__':
40.     # Tell mininet to print useful information
41.     setLogLevel('info')
42.     runExperiment()
```

Slika 3: Python skripta za kreiranje mrežne topologije i provjeru konektivnosti u Mininetu.

MININET PYTHON KLASE

Mininet Python API uključuje veliki broj Python klasa, kao što su Topo, Mininet, Host, Switch, Link i njihove podklase. API se sastoji iz tri primarna nivoa koji pružaju različite nivoe apstrakcije prilikom kreiranja emulacije:

1. *Low-level* API: Uključuje osnovne klase za mrežne uređaje i linkove, kao što su Host, Switch i Link.
2. *Mid-level* API: Sadrži veliki broj funkcija koje se koriste prilikom kreiranja mrežne topologije, kao što su: addHost(), addSwitch(), and addLink().
3. *High-level* API: Sastoji se od klase Topo koja omogućava kreiranje parametarizovanih šablona za topologije. Ovi šabloni mogu se koristiti kao argument *mn* komande (pomoću `--custom` opcije) iz komandne linije.

Kod na Slikama 2 I 3 je primjer korišćenja *high-level* API-ja. Može se primijetiti da *high-level* API pruža objektno orjentisano okruženje koje se bazira na Topo klasi. Na Slici 4 je prikazan primjer korišćenja *low-level* API-ja prilikom kreiranja topologije sa Slike 1. Sačuvajte prikazani kod u fajl *myfirsttopo_lowlevel.py* i testirajte komandom:

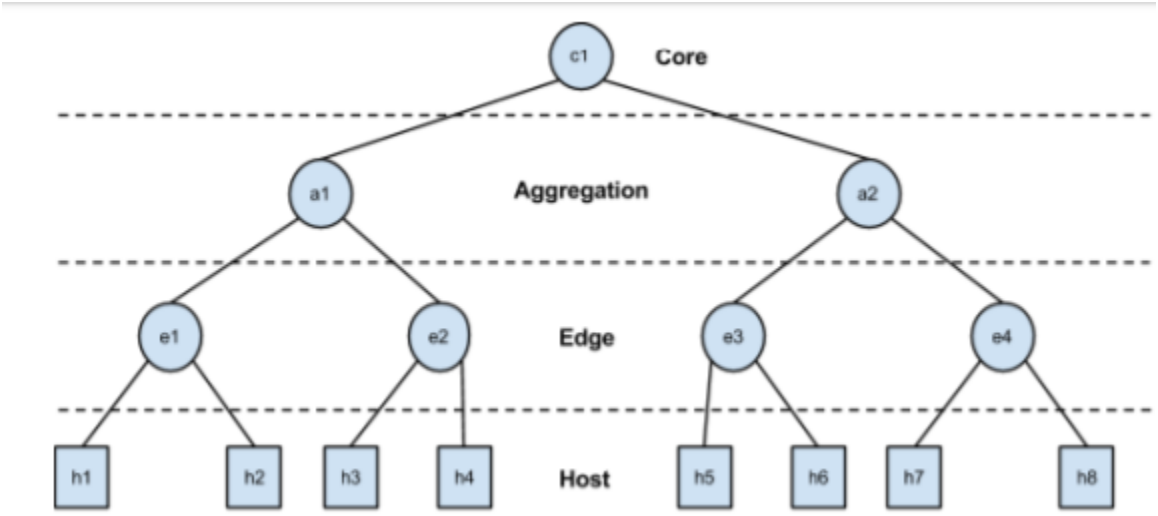
```
mininet> sudo python myfirsttopo_lowlevel.py
```

```
1.  #!/usr/bin/python
2.
3.  from mininet.node import Host,OVSSwitch,Controller
4.  from mininet.link import Link
5.
6.  h1 = Host( 'h1' )
7.  h2 = Host( 'h2' )
8.  h3 = Host( 'h3' )
9.  h4 = Host( 'h4' )
10. s1 = OVSSwitch( 's1', inNamespace=False )
11. s2 = OVSSwitch( 's2', inNamespace=False )
12. c0 = Controller( 'c0', inNamespace=False )
13. Link( h1, s1 )
14. Link( h2, s1 )
15. Link( h3, s2 )
16. Link( h4, s2 )
17. Link( s1, s2 )
18. h1.setIP( '10.0.0.1/24' )
19. h2.setIP( '10.0.0.2/24' )
20. h3.setIP( '10.0.0.3/24' )
21. h4.setIP( '10.0.0.4/24' )
22. c0.start()
23. s1.start( [ c0 ] )
24. s2.start( [ c0 ] )
25. print h1.IP
26. print h2.IP
27. print h3.IP
28. print h4.IP
29. print 'Pinging ...'
30. print h1.cmd( 'ping -c3 ', h2.IP() )
31. print h1.cmd( 'ping -c3 ', h3.IP() )
32. s1.stop()
33. s2.stop()
34. c0.stop()
```

Slika 4: Primjer korišćenja *low-level* API-ja za kreiranje toplogije sa Slike 1.

KREIRANJE STABLO TOPOLOGIJE

Mnogi Data centri koriste stablo mrežnu topologiju. Krajnji hostovi (npr. serveri) povezuju se na Top-of-Rack (ToR) *switch*-eve na ivici mreže (eng. *edge*) koji formiraju listove stabla. Jedan ili više *switch*-eva u jezgru mreže (eng. *core*) formiraju korijen stable, dok *switch*-evi u nivou agregacije između ivice i jezgra mreže čine među-čvorove stabla. Jednostavan model stablo topologije sadrži samo jedan *switch* u jezgru koji se povezuje na *n* agregacionih *switch*-eva, od kojih se svaki dalje povezuje na *n edge switch*-eva. Na svaki od *edge switch*-eva povezuje se *n* hostova (servera). Na Slici 5 prikazan je primjer stablo topologije za n=2.



Slika 5: Primjer stablo topologije.

Zadatak 5: Napisati Python program koji koristi high-level Mininet API i kreira stablo topologiju. Program treba da uzima vrijednost parametra *n* kao ulaz. Imenujte mrežne elemente u skladu sa konvencijom prikazanom na Slici 5. Organizujte čitav kod u jedan fajl naziva *simpletree_lowlevel.py* i arhivirajte. Kreiranu arhivu potrebno je poslati na mejl predmetnog nastavnika i asistenta.

Zdatak 6: Iskoristite kod napravljen u prethodnom koraku za kreiranje stablo topologije sa Slike 5. Provjerite konektivnost svih elemenata u mreži i zapišite vaša zapažanja. Stopirajte *core switch* c1 komandom: `c1 stop`. Ponovo provjerite konektivnost između hostova i objasnite nova zapažanja.

KORISNA LITERATURA

Uvod u Mininet: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet
Mininet tutorijal: http://mininet.org/walkthrough/
Mininet Python API upustvo: http://mininet.org/api/annotated.html