LAPORAN ADMINISTRASI DESAIN DAN JARINGAN

MININET, MININAM, OPENFLOW, DAN POXCONTROLLER

DISUSUN OLEH

FITRIA ADINE YASMINE BELINDA	161402120
DESSY ATIKA	161402060
YASMIN NABILAH DETA	161402078
AMANDA NOVIANDHINI	161402111
TALITHA AZURA PUTRI AULIA	161402087

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

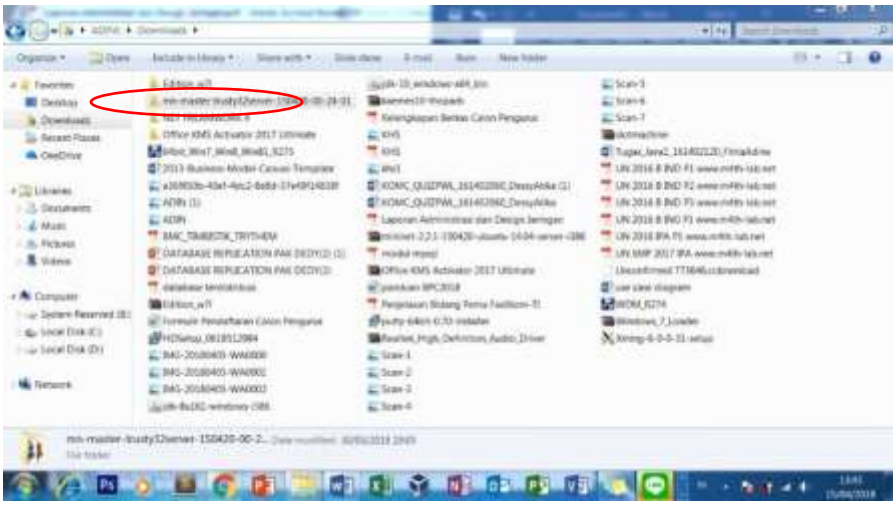
2018

MININET

Mininet adalah emulator berbasis CLI yang digunakan untuk membuat sebuah topologi jaringan pada Software Defined Network. Pada Mininet, sudah terdapat beberapa topologi bawaan yang dapat langsung digunakan dengan menggunakan perintah (command) tertentu. Beberapa topologi bawaan tersebut antara lain topologi single, tree dan linear. Berikut adalah tutorial instalasi Mininet:

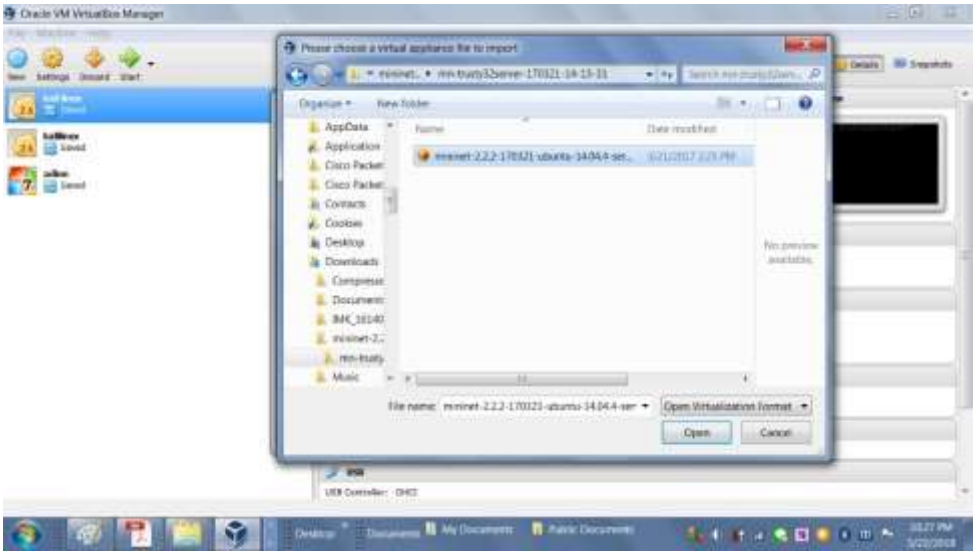
1. Download Mininet

   Kita dapat mendownload secara gratis Mininet melalui http://mininet.org/



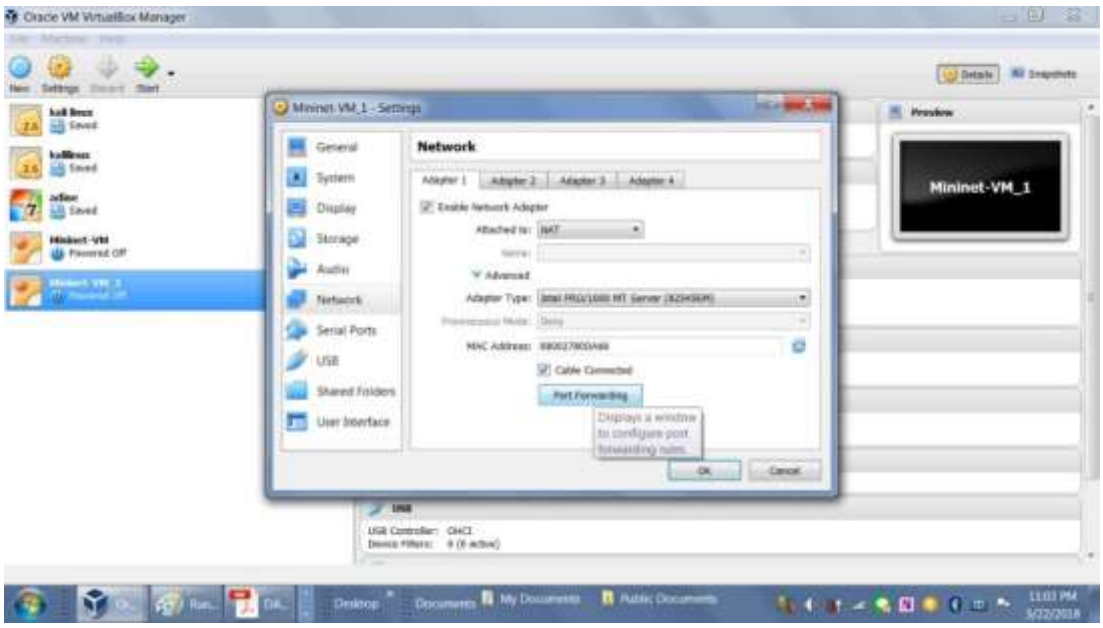
2. Import Mininet kedalam Virtual Machine(VM)

   Kita pilih *file* lalu klik *import appliance* dan pilih folder kita tempatkan file mininet tadi setelah itu *open*.

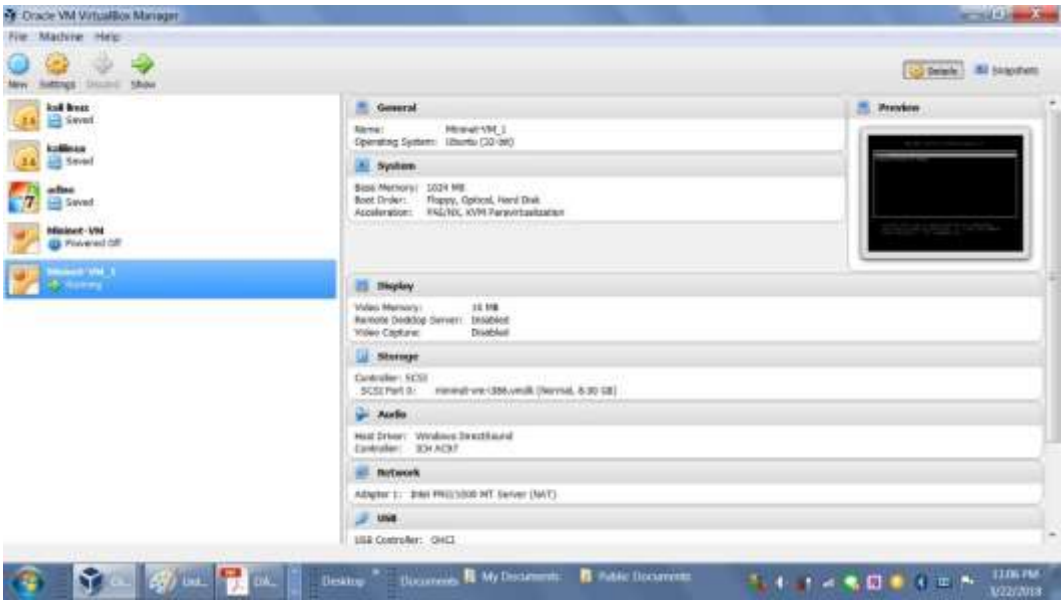


3. Melakukan konfigurasi akses

   Dalam hal ini kita akan melakukan post forwarding caraya adalah klik *setting* lalu pilih *networks* dan pada kotak dialog pilih dan klik *post forwaring*. Setelah itu tambahkan guest port dan host port.

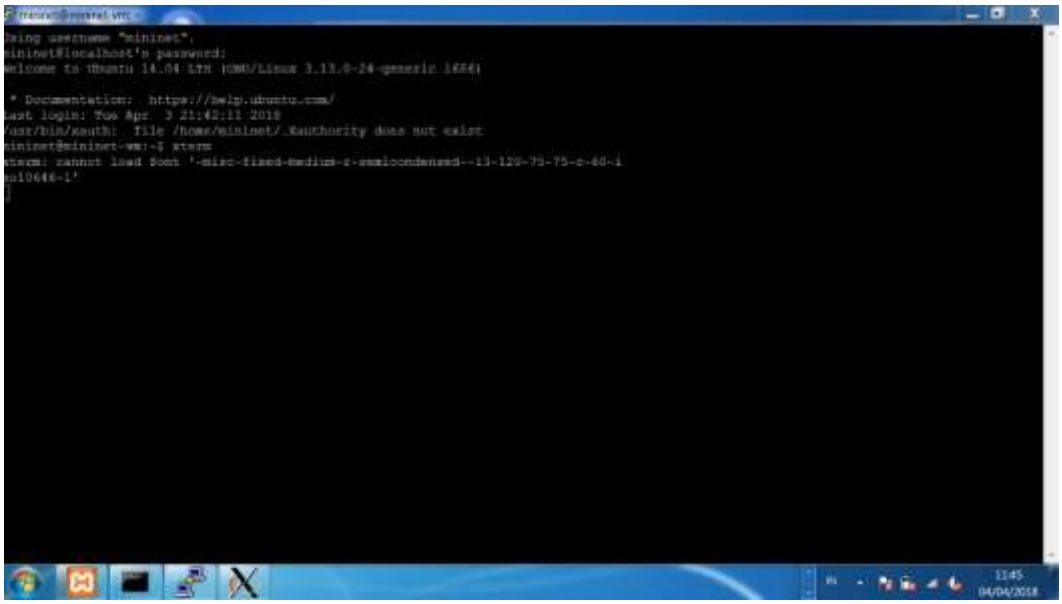


4. Jalankan VM



5. Xming

   Xming adalah sebuah server yang digunakan untuk mengakses ssh untuk sistem operasi windows.

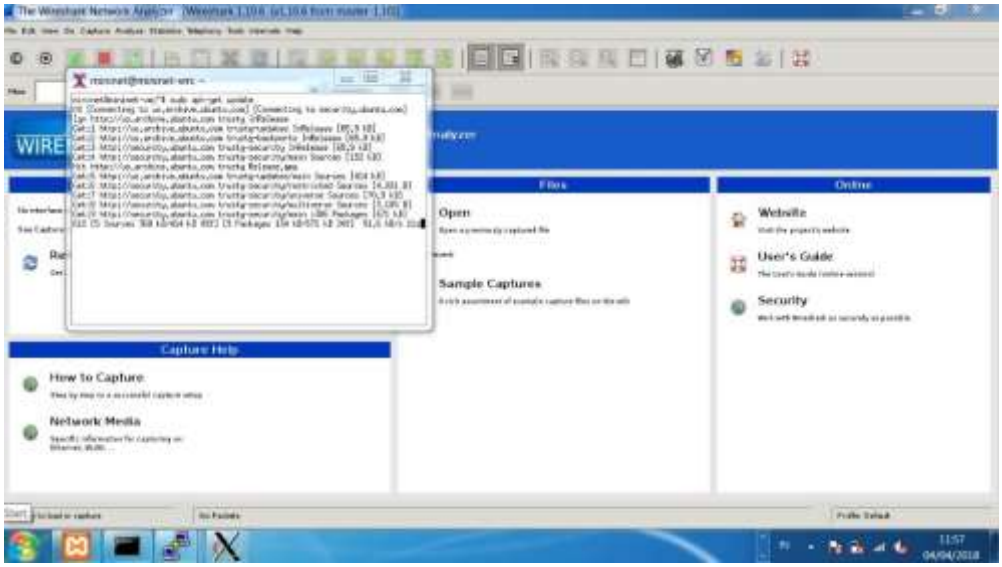


6. SSH to VM

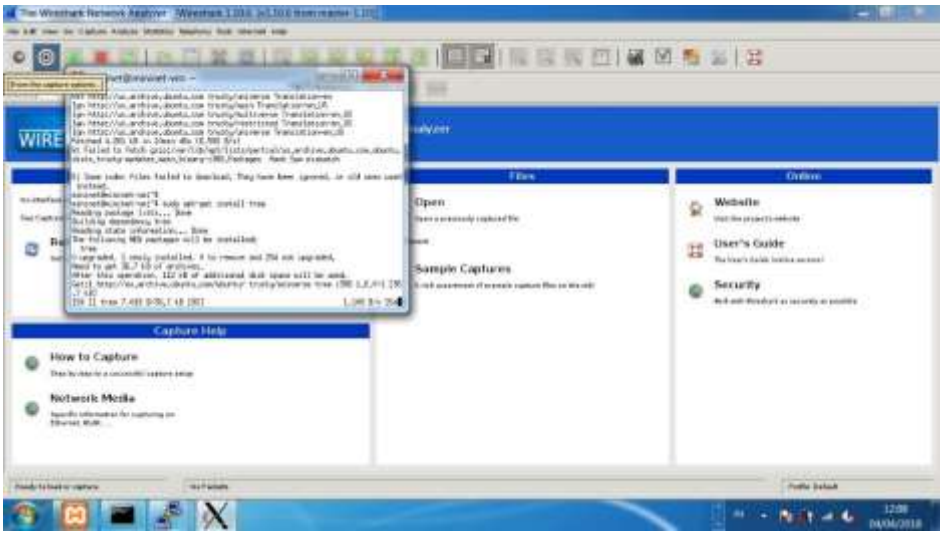
   SSH yang kami gunakan adalah PuTTY. Pertama kita buka command prompt dan masukkan kedalam direktori tempat kita menaruh folder PuTTY dan masukkan perintah putty.exe -X -P 2222 -1 mininet localhost. Apabila berhasil maka akan muncul terminal baru dengan background putih(Xterminal).

7. Access via SSH

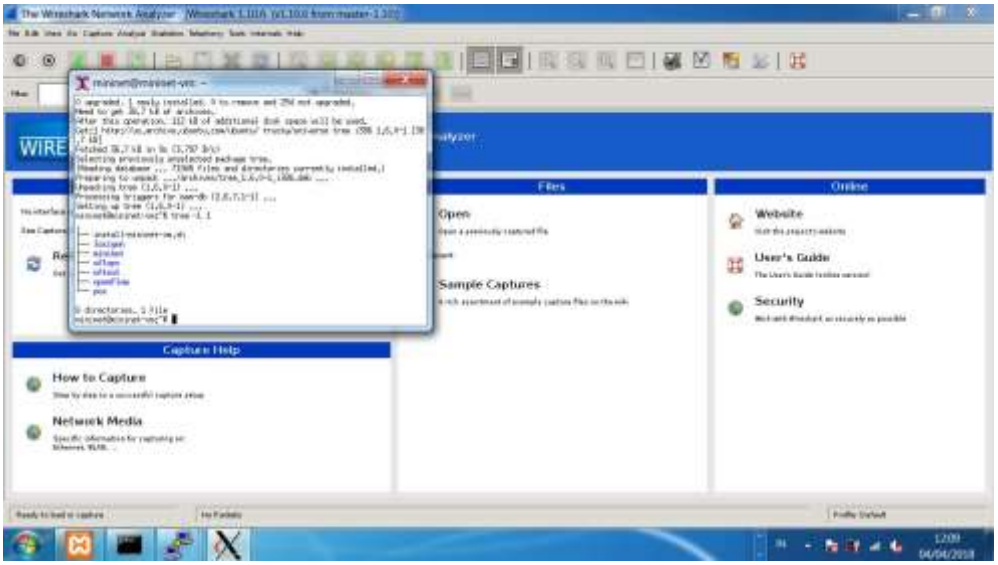
8. Melakukan Test Wireshark



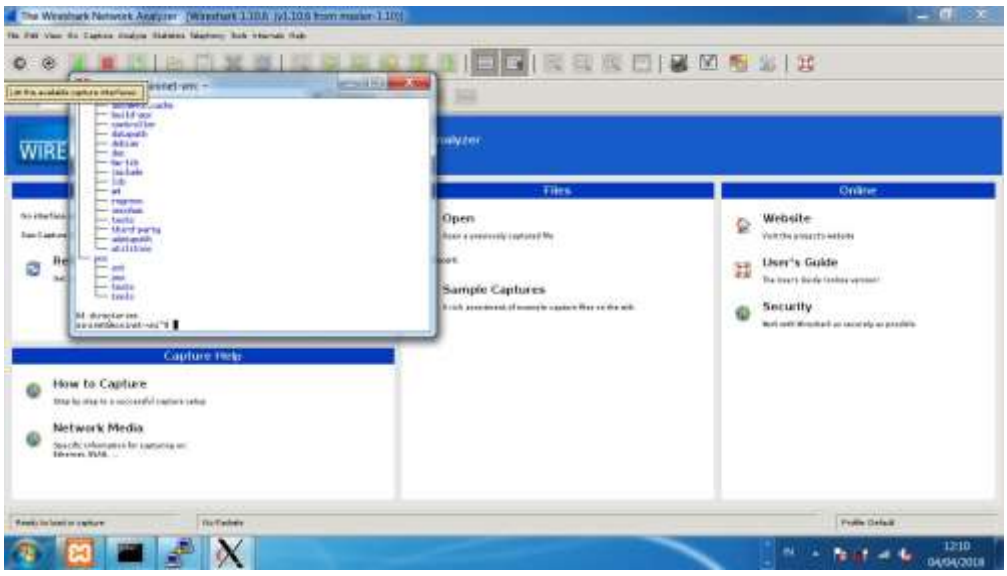
9. Melakukan instalasi tree

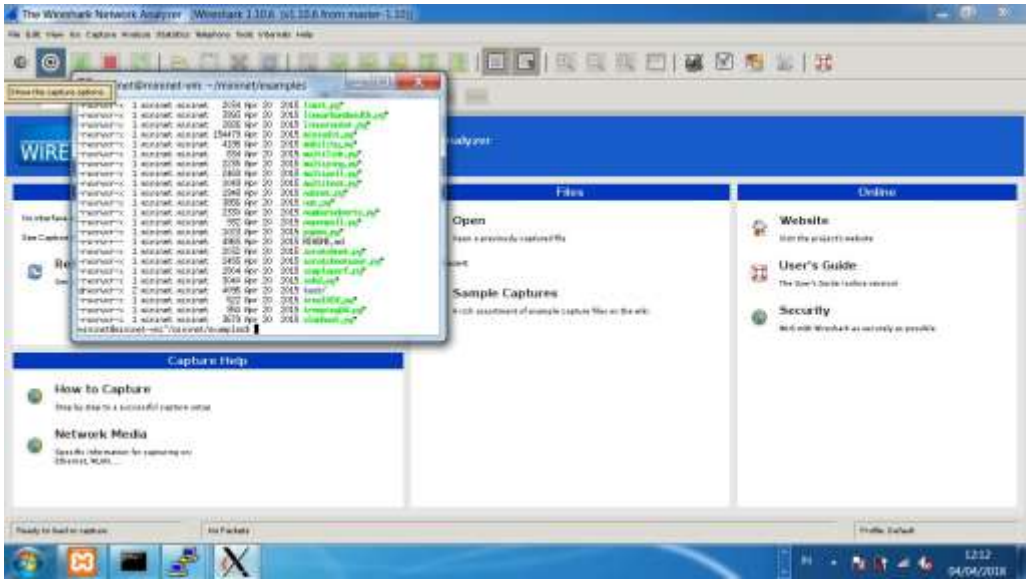


10. Mengupdate tree

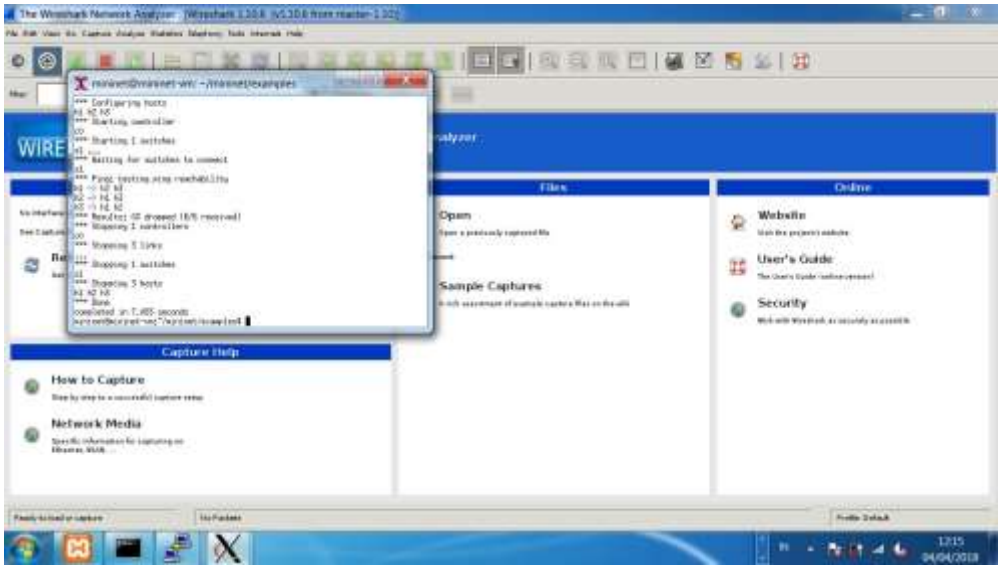


11. Pemeriksaan pada mininet tree

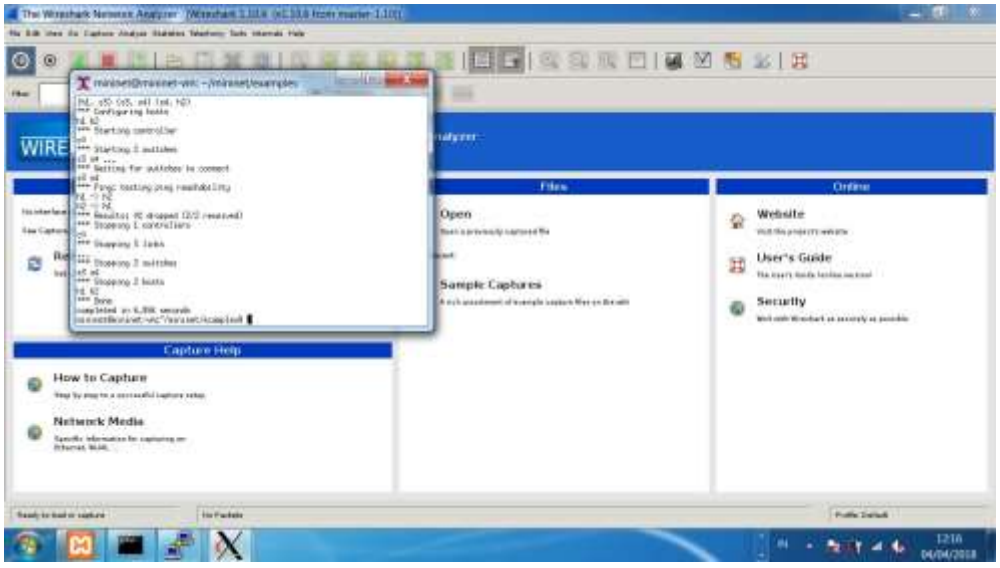




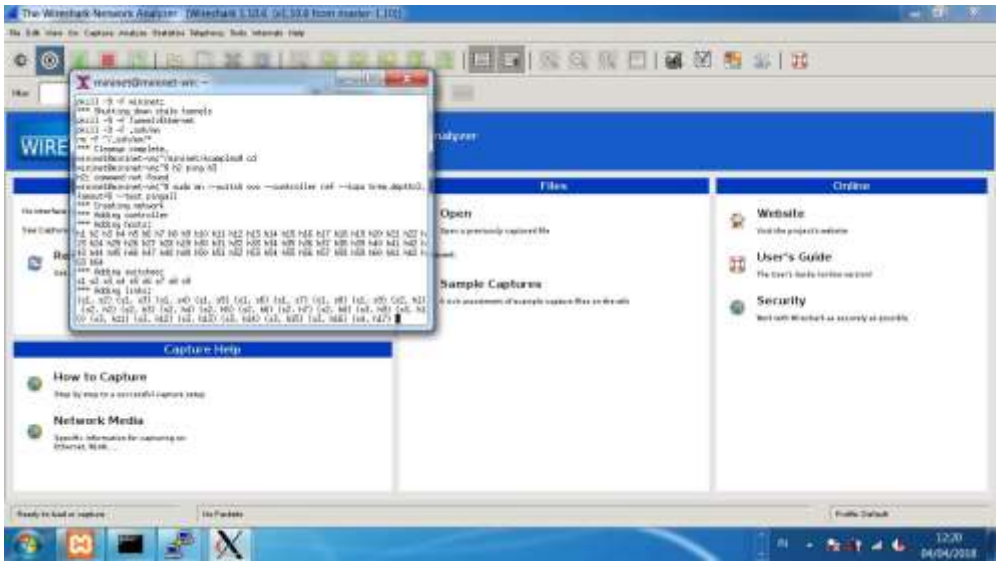
12. Melakukan link TC



13. Melakukan ping



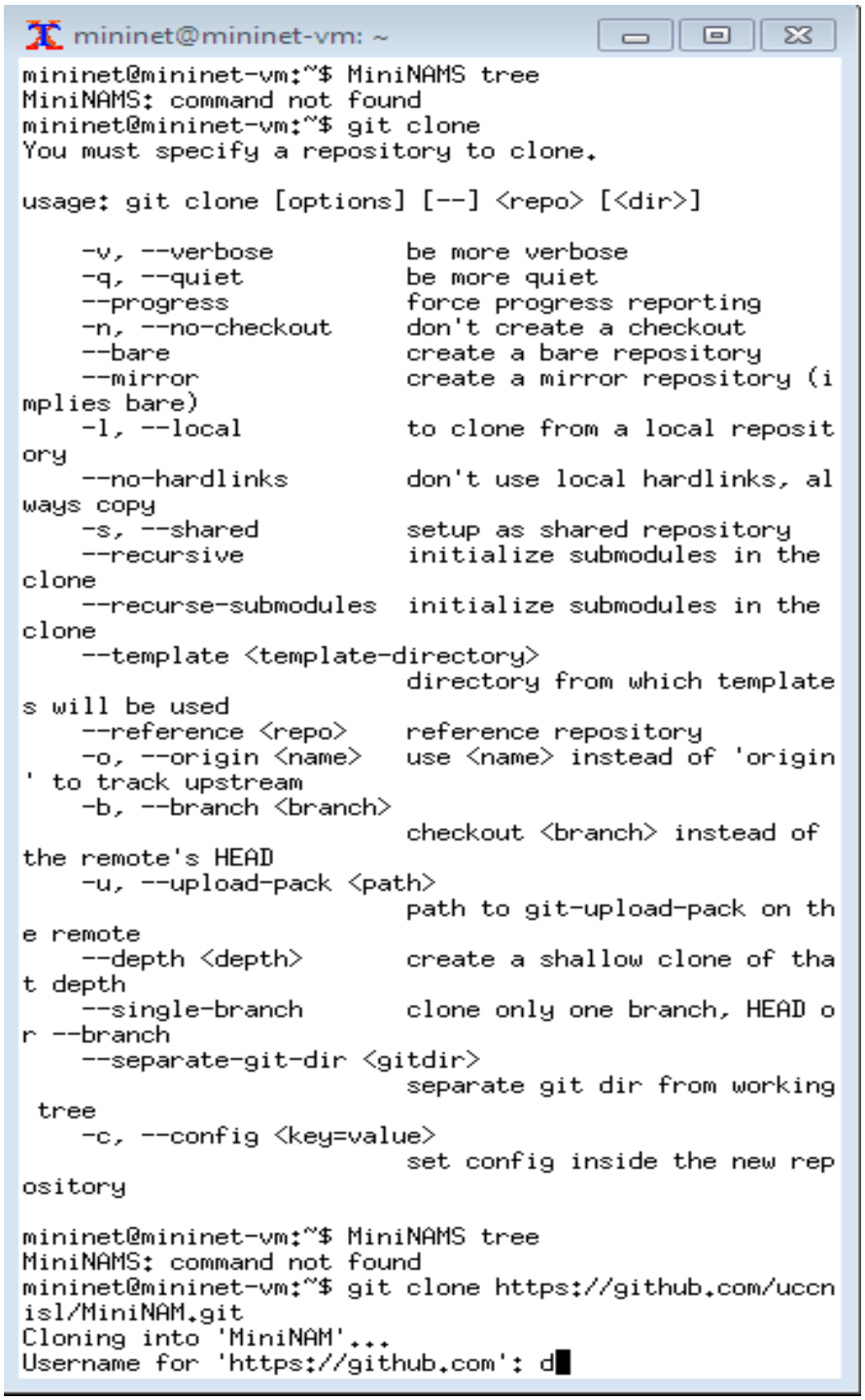
14. Create new network



MININAM

Melanjutkan proses dari mininet,Tahap-tahap dalam mininam sebagai berikut:

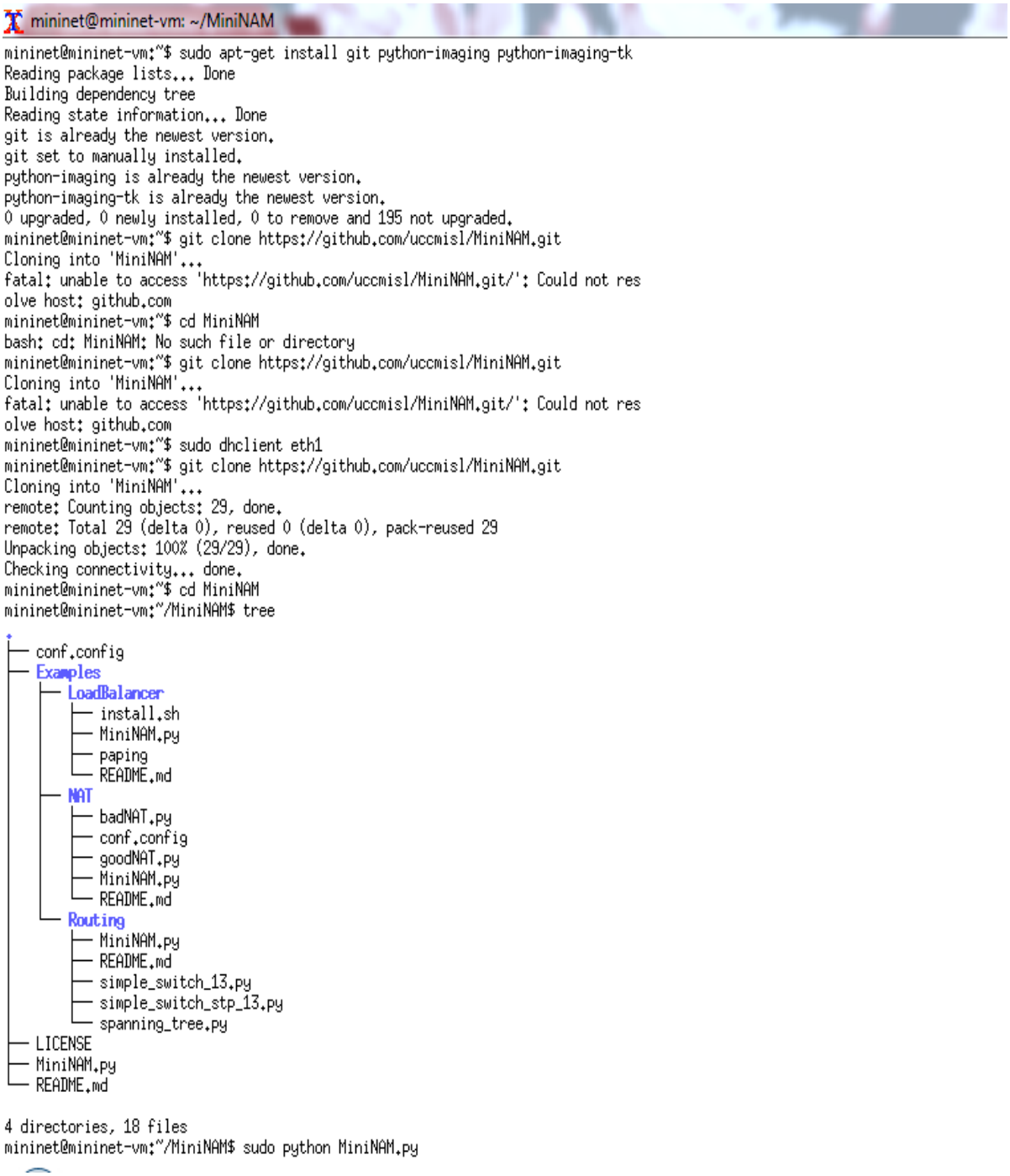
1. Melakukan cloning untuk masuk ke MiniNAM



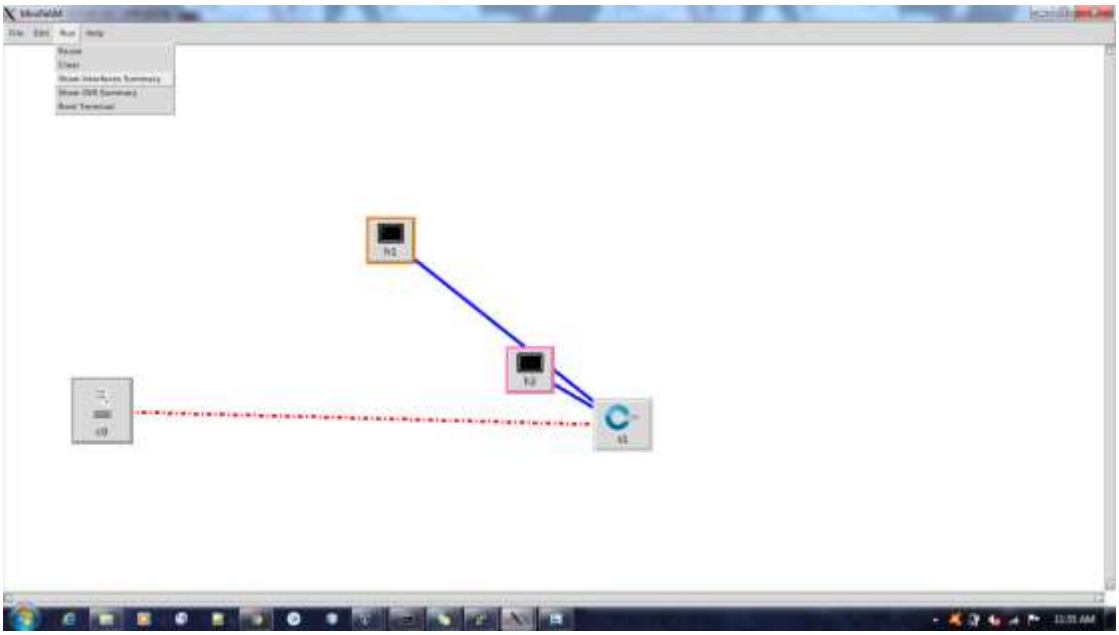
2. Melakukan konfigurasi host h1 dan h2

```
mininet@mininet-vm: ~/MiniNAM
Cloning into 'MiniNAM'...
fatal: unable to access 'https://github.com/uccmisl/MiniNAM.git/': Could not res
olve host: github.com
mininet@mininet-vm:~$ sudo dhclient eth1
mininet@mininet-vm:~$ git clone https://github.com/uccmisl/MiniNAM.git
Cloning into 'MiniNAM'...
remote: Counting objects: 29, done.
remote: Total 29 (delta 0), reused 0 (delta 0), pack-reused 29
Unpacking objects: 100% (29/29), done.
Checking connectivity... done.
mininet@mininet-vm:~$ cd MiniNAM
mininet@mininet-vm:~/MiniNAM$ tree
.
├── conf.config
├── Examples
│   ├── LoadBalancer
│   │   ├── install.sh
│   │   ├── MiniNAM.py
│   │   ├── paping
│   │   └── README.md
│   ├── NAT
│   │   ├── badNAT.py
│   │   ├── conf.config
│   │   ├── goodNAT.py
│   │   ├── MiniNAM.py
│   │   └── README.md
│   └── Routing
│       ├── MiniNAM.py
│       ├── README.md
│       ├── simple_switch_13.py
│       ├── simple_switch_stp_13.py
│       └── spanning_tree.py
├── LICENSE
├── MiniNAM.py
└── README.md

4 directories, 18 files
mininet@mininet-vm:~/MiniNAM$ sudo python MiniNAM.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

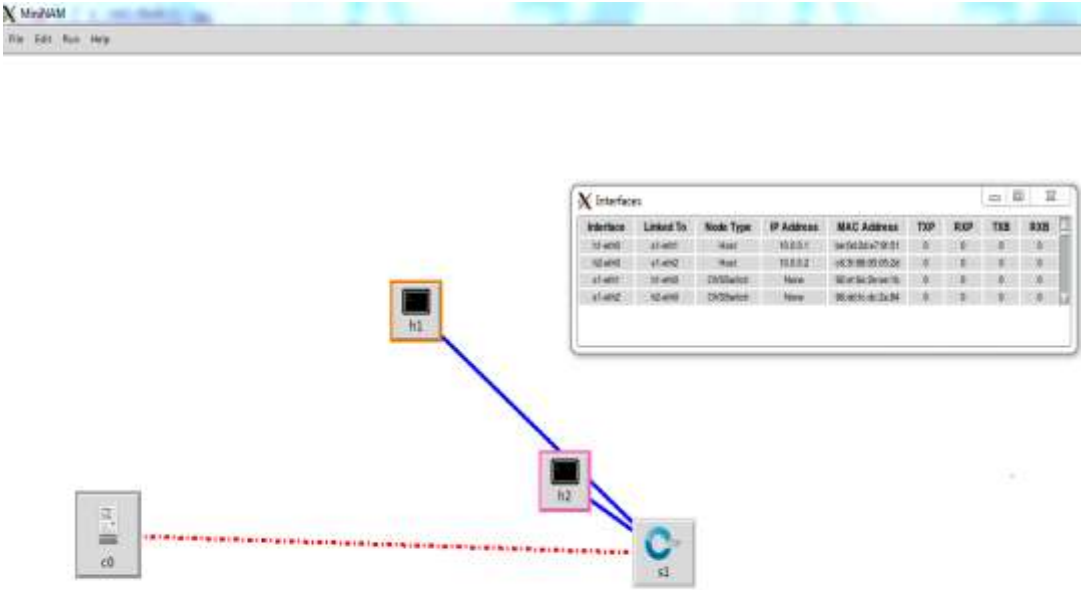
2. Ketik “Sudo Pyton MiniNAM.py”



4. Akan muncul gambar seperti dibawah ini dalam Xming dan klik “Show Interfaces Summary”.



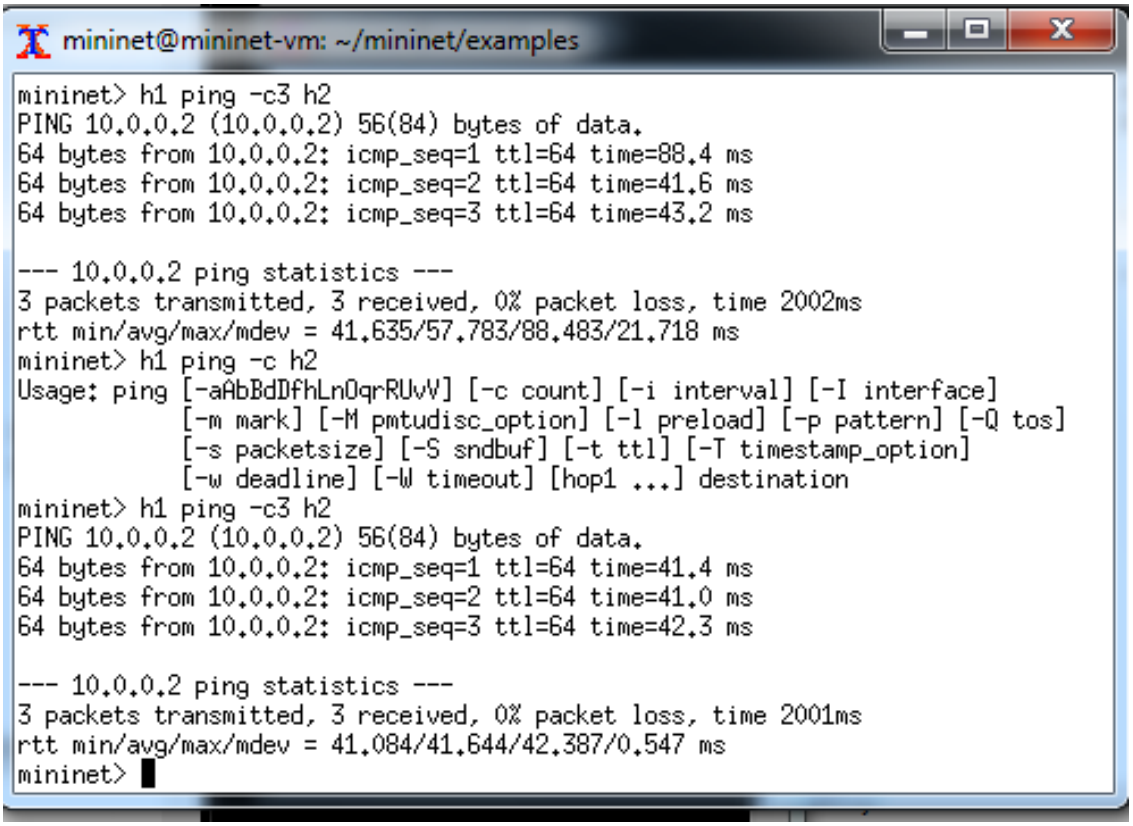
5.Setelah meng-Klik “ Show interfaces summery” akan muncul interface seperti gambar dibawah ini:



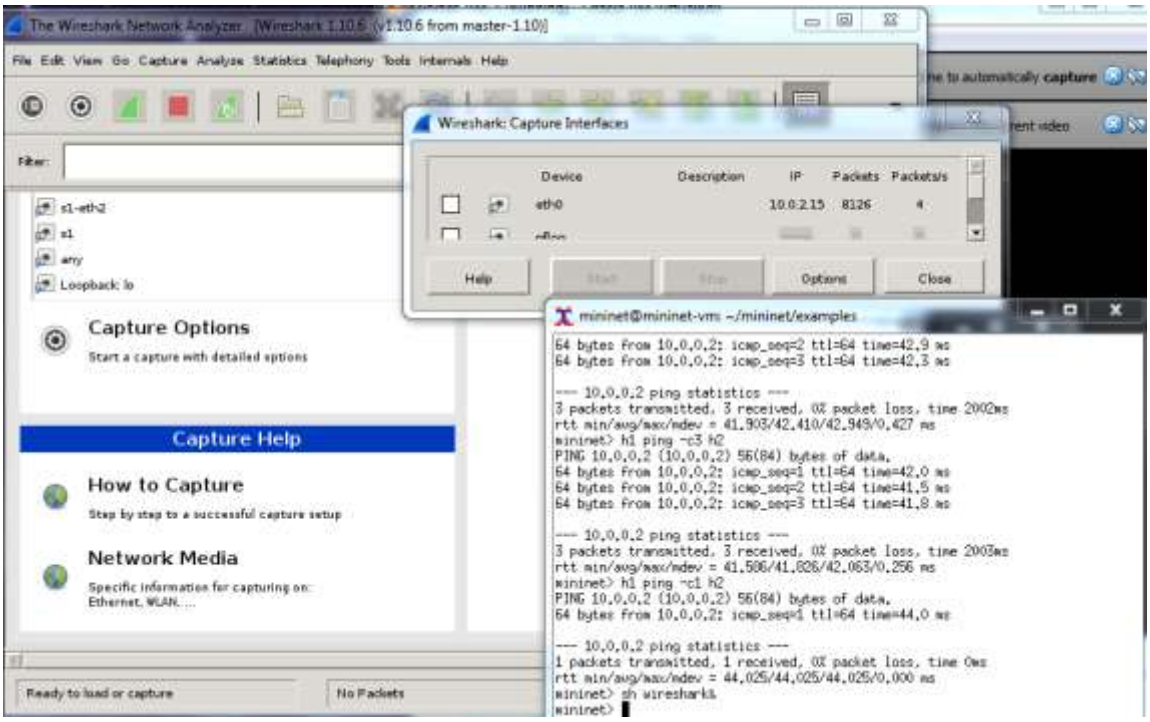
OPEN FLOW

OpenFlow adalah protokol yang memungkinkan pengaturan penjaluran dan pengiriman oscket ketika melalui sebuah switch.Setiap switch hanya berfungsi meneruskan packet yang lewat ke port yang sesuai tanpa dapat membedakan type protokol data yang dikirimkan.Pada openflow kita tidak hanya dapat melakukan flow forwarding berbasis netwrok layer tetapi juga dapat dilakukan pengaturan pergerakan paket secara terpusat mulai dari layer 2 sampe layer 7 forwarding*(flow granuality)*,sehingga aliran paket di jaringan dapat diprogram secara independen.

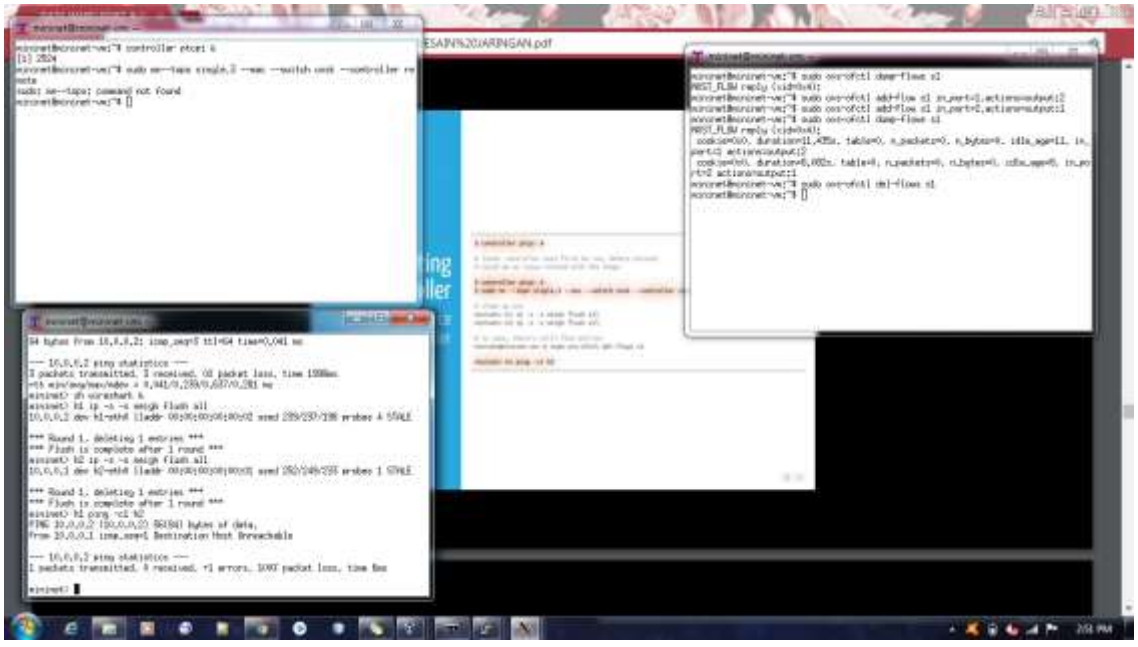
1. Manual Entry Flow



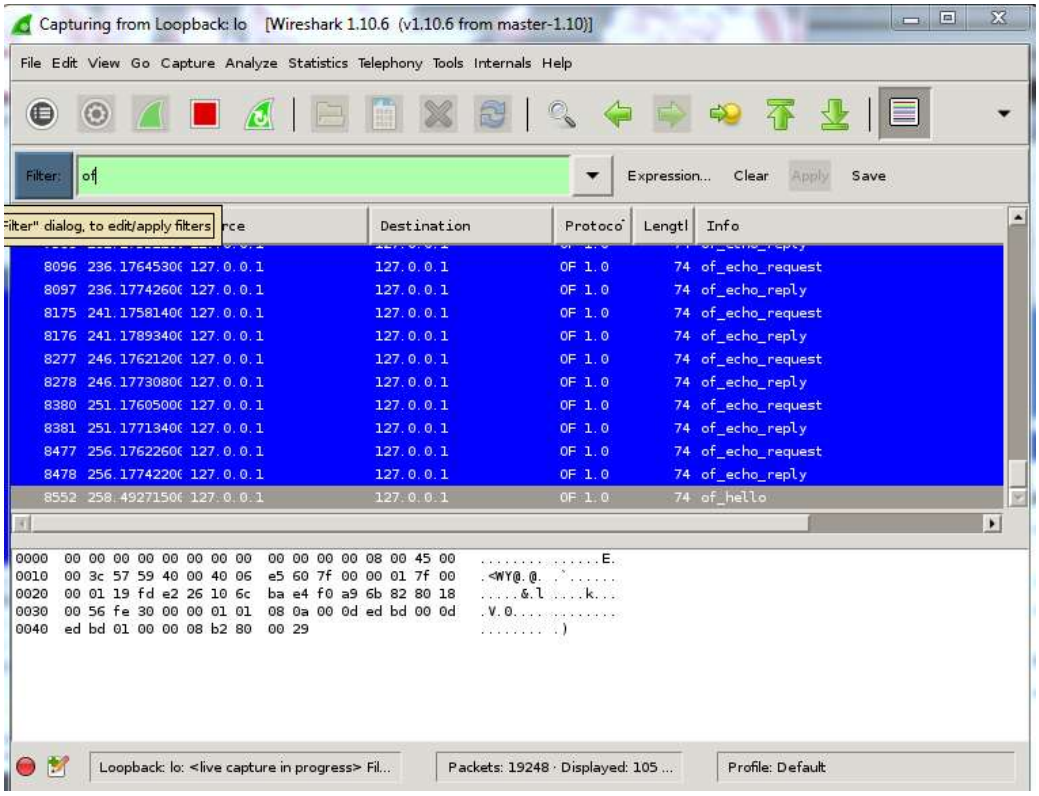
1. Membuka capture interface



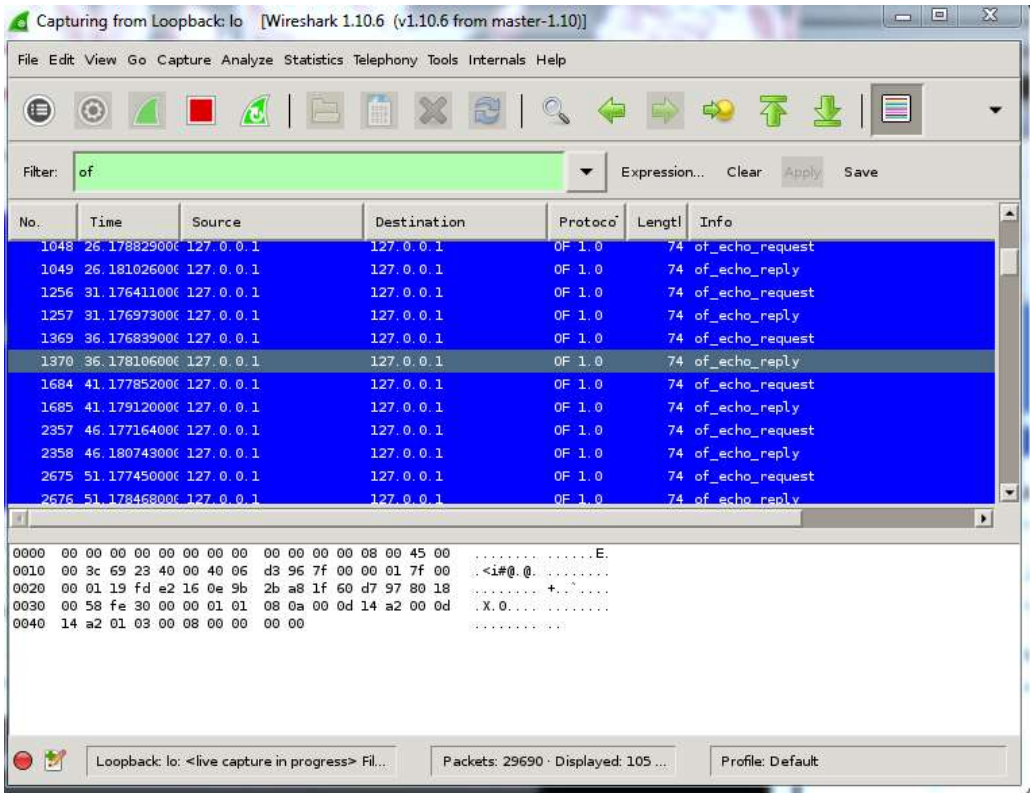
STARTING CONTROLLER



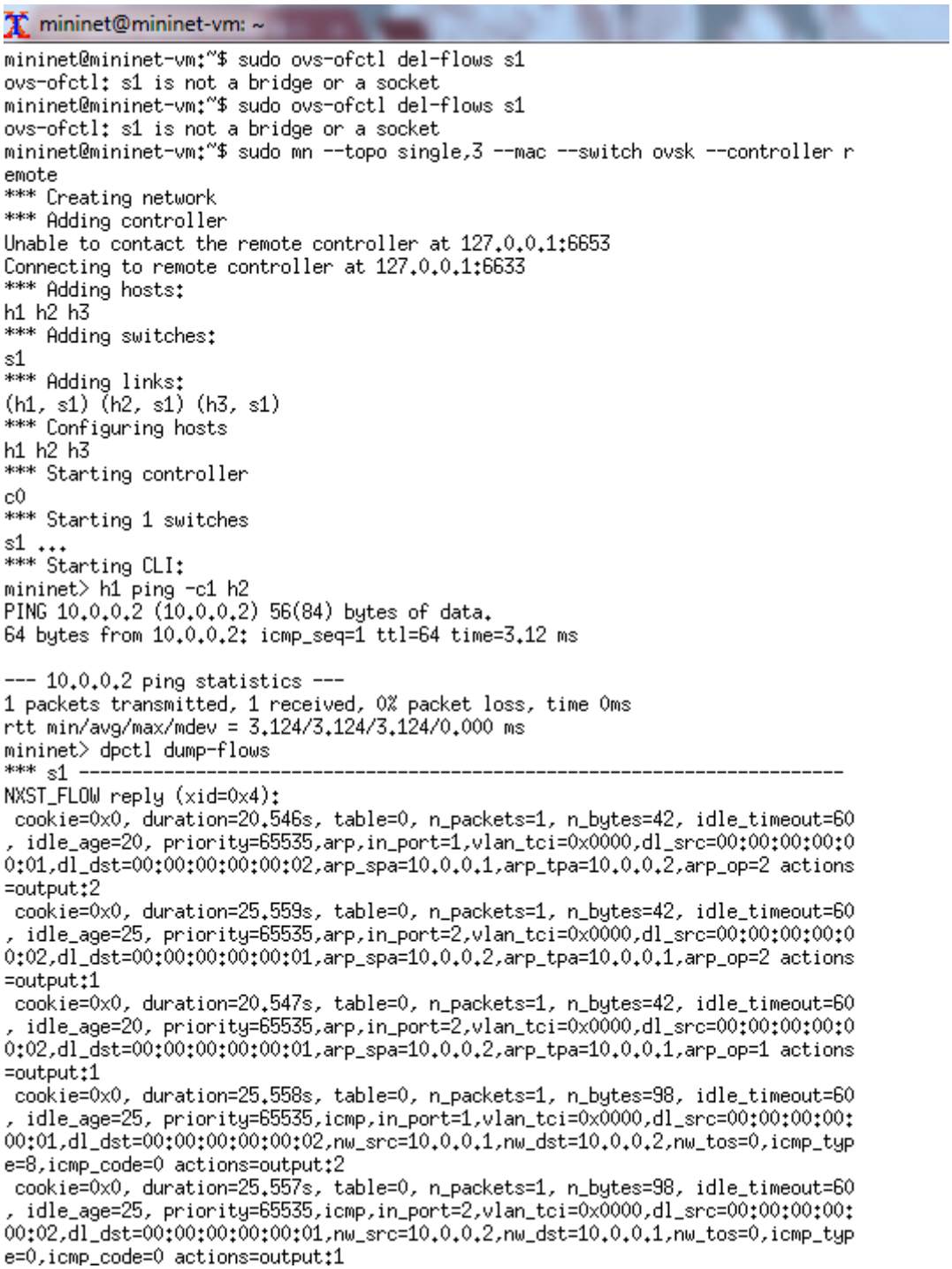
2. Filter *of* pada Wireshark kemudian apply.



3. Inspect *packages.*

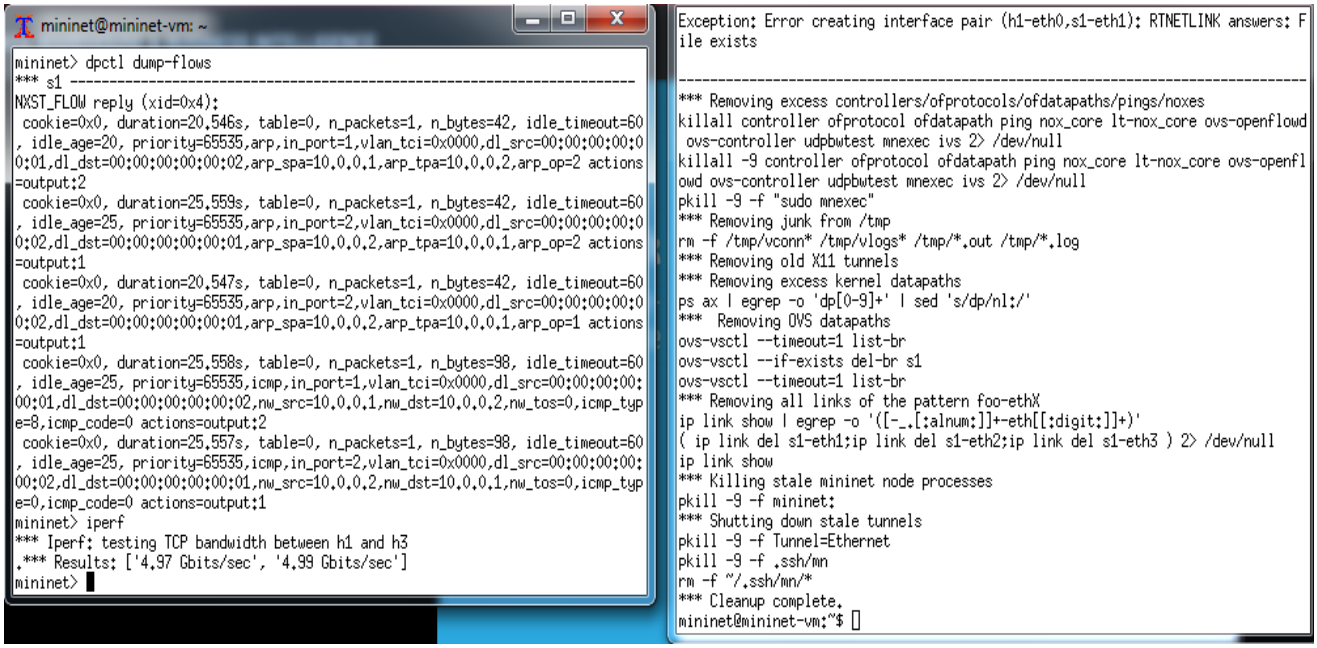


4. Flow Entries dengan mengetik “dpctl dump-flows”.



```
mininet@mininet-vm: ~
mininet@mininet-vm:~$ sudo ovs-ofctl del-flows s1
ovs-ofctl: s1 is not a bridge or a socket
mininet@mininet-vm:~$ sudo ovs-ofctl del-flows s1
ovs-ofctl: s1 is not a bridge or a socket
mininet@mininet-vm:~$ sudo mn --topo single,3 --mac --switch ovsk --controller r
emote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ping -c1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=3.12 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.124/3.124/3.124/0.000 ms
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=20.546s, table=0, n_packets=1, n_bytes=42, idle_timeout=60
, idle_age=20, priority=65535,arp,in_port=1,vlan_tci=0x0000,dl_src=00:00:00:00:0
0:01,dl_dst=00:00:00:00:00:02,arp_spa=10.0.0.1,arp_tpa=10.0.0.2,arp_op=2 actions
=output:2
 cookie=0x0, duration=25.559s, table=0, n_packets=1, n_bytes=42, idle_timeout=60
, idle_age=25, priority=65535,arp,in_port=2,vlan_tci=0x0000,dl_src=00:00:00:00:0
0:02,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=2 actions
=output:1
 cookie=0x0, duration=20.547s, table=0, n_packets=1, n_bytes=42, idle_timeout=60
, idle_age=20, priority=65535,arp,in_port=2,vlan_tci=0x0000,dl_src=00:00:00:00:0
0:02,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=1 actions
=output:1
 cookie=0x0, duration=25.558s, table=0, n_packets=1, n_bytes=98, idle_timeout=60
, idle_age=25, priority=65535,icmp,in_port=1,vlan_tci=0x0000,dl_src=00:00:00:00:
00:01,dl_dst=00:00:00:00:00:02,nw_src=10.0.0.1,nw_dst=10.0.0.2,nw_tos=0,icmp_typ
e=8,icmp_code=0 actions=output:2
 cookie=0x0, duration=25.557s, table=0, n_packets=1, n_bytes=98, idle_timeout=60
, idle_age=25, priority=65535,icmp,in_port=2,vlan_tci=0x0000,dl_src=00:00:00:00:
00:02,dl_dst=00:00:00:00:00:01,nw_src=10.0.0.2,nw_dst=10.0.0.1,nw_tos=0,icmp_typ
e=0,icmp_code=0 actions=output:1
```

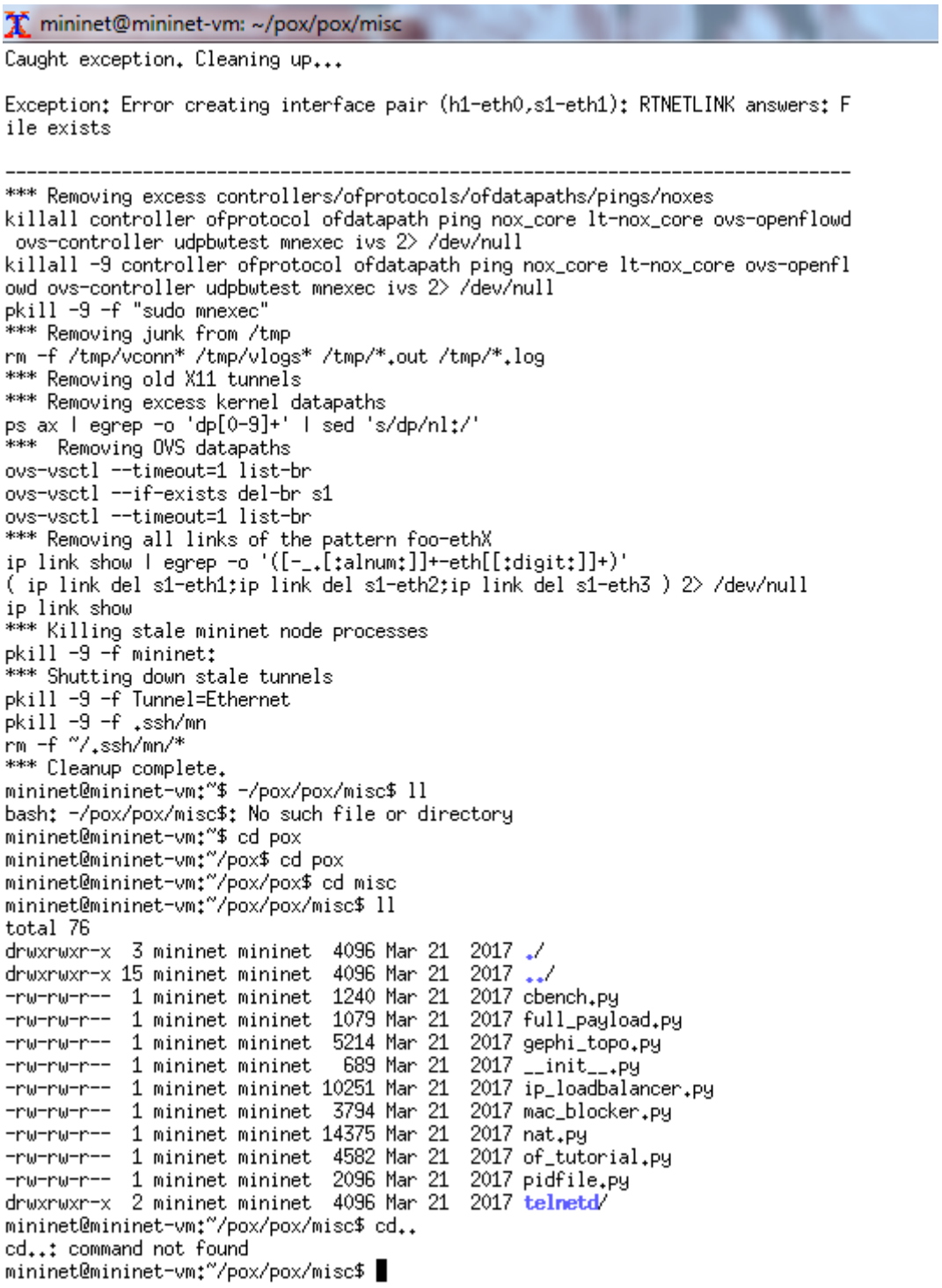
5. Benchmark kernel-v vs –user space



POX CONTROLLER

Pox Controller merupakan sebuah platform pengembangan *open source* untuk aplikasi *Software Developed Network* (SDN) berdasarkan pada bahasa pemograman Python merupakan kontroler OpenFlow. POX memungkinkan proses perancangan dan pemangunan jaringan yag lebih cepat,serta lebih menjadi umum digunakan pendahulunya NOX.

1. File Pox Controller dalam Mininet.



```
mininet@mininet-vm:~/pox/pox/misc$ ll
total 76
drwxrwxr-x  3 mininet mininet  4096 Mar 21  2017 ./
drwxrwxr-x 15 mininet mininet  4096 Mar 21  2017 ../
-rw-rw-r--  1 mininet mininet  1240 Mar 21  2017 cbench.py
-rw-rw-r--  1 mininet mininet  1079 Mar 21  2017 full_payload.py
-rw-rw-r--  1 mininet mininet  5214 Mar 21  2017 gephi_topo.py
-rw-rw-r--  1 mininet mininet   689 Mar 21  2017 __init__.py
-rw-rw-r--  1 mininet mininet 10251 Mar 21  2017 ip_loadbalancer.py
-rw-rw-r--  1 mininet mininet  3794 Mar 21  2017 mac_blocker.py
-rw-rw-r--  1 mininet mininet 14375 Mar 21  2017 nat.py
-rw-rw-r--  1 mininet mininet  4582 Mar 21  2017 of_tutorial.py
-rw-rw-r--  1 mininet mininet  2096 Mar 21  2017 pidfile.py
drwxrwxr-x  2 mininet mininet  4096 Mar 21  2017 telnetd/
mininet@mininet-vm:~/pox/pox/misc$ cd..
cd..: command not found
mininet@mininet-vm:~/pox/pox/misc$ cd ..
mininet@mininet-vm:~/pox/pox$ cd forwarding
mininet@mininet-vm:~/pox/pox/forwarding$ ll
total 96
drwxrwxr-x  2 mininet mininet  4096 Mar 21  2017 ./
drwxrwxr-x 15 mininet mininet  4096 Mar 21  2017 ../
-rw-rw-r--  1 mininet mininet  1092 Mar 21  2017 hub.py
-rw-rw-r--  1 mininet mininet   651 Mar 21  2017 __init__.py
-rw-rw-r--  1 mininet mininet  4426 Mar 21  2017 l2_flowvisor.py
-rw-rw-r--  1 mininet mininet  6692 Mar 21  2017 l2_learning.py
-rw-rw-r--  1 mininet mininet 15558 Mar 21  2017 l2_multi.py
-rw-rw-r--  1 mininet mininet  4324 Mar 21  2017 l2_nx.py
-rw-rw-r--  1 mininet mininet  2105 Mar 21  2017 l2_nx_self_learning.py
-rw-rw-r--  1 mininet mininet  2882 Mar 21  2017 l2_pairs.py
-rw-rw-r--  1 mininet mininet 12330 Mar 21  2017 l3_learning.py
-rw-rw-r--  1 mininet mininet 14102 Mar 21  2017 topo_proactive.py
mininet@mininet-vm:~/pox/pox/forwarding$
```

2. Memilih of_tutorial.py pada **pox/ppox/misc**.

mininet@mininet-vm: ~/pox/pox/misc

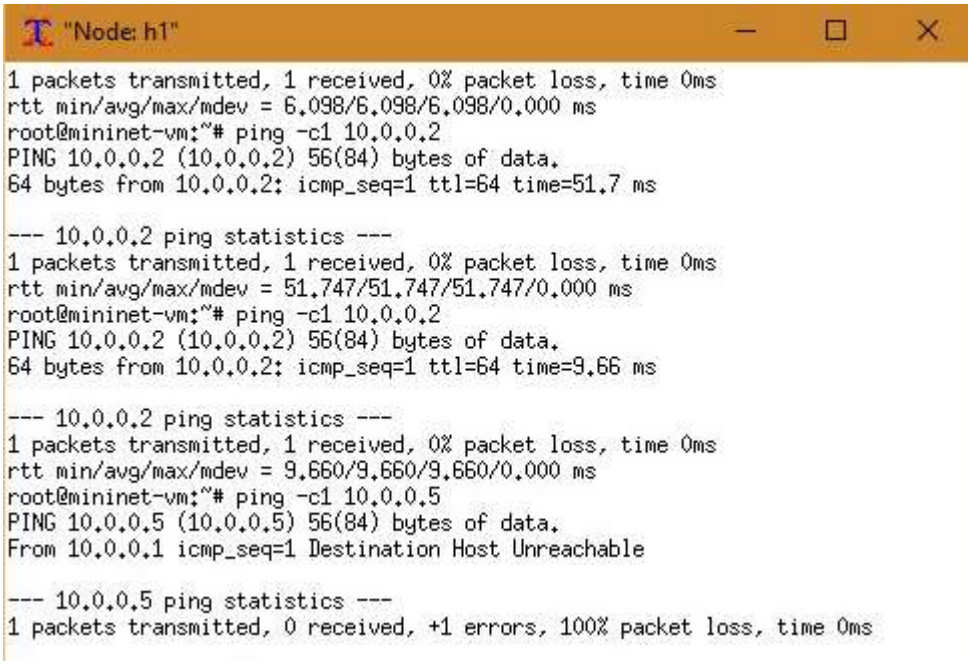
```
from pox.core import core
import pox.openflow.libopenflow_01 as of

log = core.getLogger()


class Tutorial (object):
  """
  A Tutorial object is created for each switch that connects.
  A Connection object for that switch is passed to the __init__ function.
  """
  def __init__ (self, connection):
    # Keep track of the connection to the switch so that we can
    # send it messages!
    self.connection = connection

    # This binds our PacketIn event listener
    connection.addListeners(self)

    # Use this table to keep track of which ethernet address is on
    # which switch port (keys are MACs, values are ports).
    self.mac_to_port = {}


  def resend_packet (self, packet_in, out_port):
    """
    Instructs the switch to resend a packet that it had sent to us.
    "packet_in" is the ofp_packet_in object the switch had sent to the
    controller due to a table-miss.
    """
    msg = of.ofp_packet_out()
    msg.data = packet_in

    # Add an action to send to the specified port
    action = of.ofp_action_output(port = out_port)
    msg.actions.append(action)

    # Send message to switch
    self.connection.send(msg)


  def act_like_hub (self, packet, packet_in):
    """
    Implement hub-like behavior -- send all packets to all ports besides
    the input port.
    """

    # We want to output to all ports -- we do that using the special
    # OFPP_ALL port as the output port.  (We could have also used
    # OFPP_FLOOD.)
    self.resend_packet(packet_in, of.OFPP_ALL)

    # Note that if we didn't get a valid buffer_id, a slightly better
    # implementation would check that we got the full data before
    # sending it (len(packet_in.data) should be == packet_in.total_len)).
```

3. Berikan perintah ‘tcpdump –XX –n –i h2-eth0> h2.txt’ pada terminal node 2 dan ‘tcpdump –XX –n –i h3-eth0> h3.txt’ pada terminal node 3.

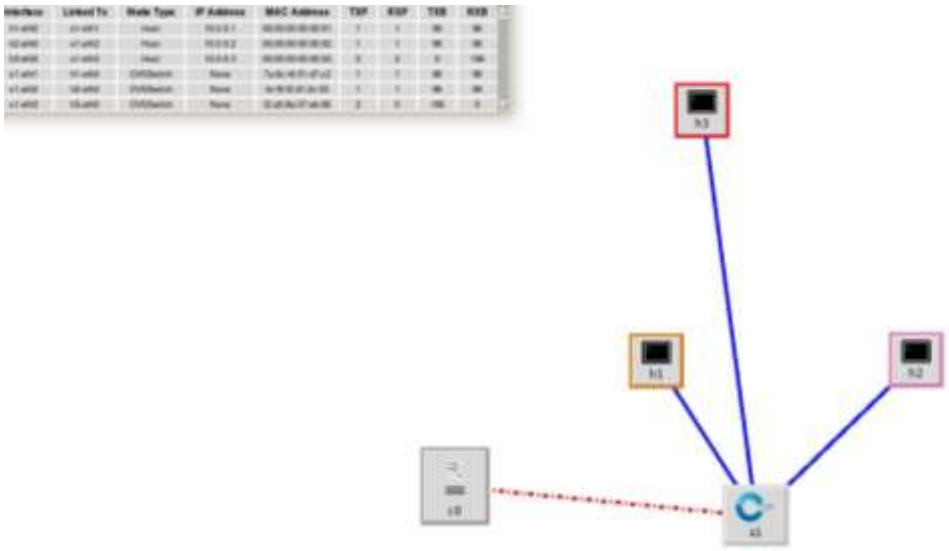
```
mininet@mininet-vm:~$ cat h2.txt
18:01:48.413946 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 2117, seq 1, length 64
        0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
        0x0010:  0054 d274 4000 4001 5432 0a00 0001 0a00  .T.t@.@.T2......
        0x0020:  0002 0800 009f 0845 0001 7c2f c05a c28d  .......E..|/.Z..
        0x0030:  0500 0809 0a0b 0c0d 0e0f 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
18:01:48.413979 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 2117, seq 1, length 64
        0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
        0x0010:  0054 1de7 0000 4001 48c0 0a00 0002 0a00  .T....@.H.......
        0x0020:  0001 0000 089f 0845 0001 7c2f c05a c28d  .......E..|/.Z..
        0x0030:  0500 0809 0a0b 0c0d 0e0f 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
18:01:53.407484 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
        0x0000:  0000 0000 0002 0000 0000 0001 0806 0001  ................
        0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
        0x0020:  0000 0000 0000 0a00 0002                 ..........
18:01:53.407513 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
        0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
        0x0010:  0800 0604 0002 0000 0000 0002 0a00 0002  ................
        0x0020:  0000 0000 0001 0a00 0001                 ..........
18:03:49.367243 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 2145, seq 1, length 64
        0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
        0x0010:  0054 fe72 4000 4001 2834 0a00 0001 0a00  .T.r@.@.(4......
        0x0020:  0002 0800 bb92 0861 0001 f52f c05a 8e7d  .......a.../.Z.}
        0x0030:  0500 0809 0a0b 0c0d 0e0f 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
18:03:49.367275 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 2145, seq 1, length 64
        0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
        0x0010:  0054 478a 0000 4001 1f1d 0a00 0002 0a00  .TG...@.........
        0x0020:  0001 0000 c392 0861 0001 f52f c05a 8e7d  .......a.../.Z.}
        0x0030:  0500 0809 0a0b 0c0d 0e0f 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
```

```
mininet@mininet-vm:~$ cat h3.txt
18:01:48.413943 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 2117, seq 1, length 64
        0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
        0x0010:  0054 d274 4000 4001 5432 0a00 0001 0a00  .T.t@.@.T2......
        0x0020:  0002 0800 009f 0845 0001 7c2f c05a c28d  .......E..|/.Z..
        0x0030:  0500 0809 0a0b 0c0d 0e0f 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
18:01:48.415713 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 2117, seq 1, length 64
        0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
        0x0010:  0054 1de7 0000 4001 48c0 0a00 0002 0a00  .T....@.H.......
        0x0020:  0001 0000 089f 0845 0001 7c2f c05a c28d  .......E..|/.Z..
        0x0030:  0500 0809 0a0b 0c0d 0e0f 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
18:01:53.407480 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
        0x0000:  0000 0000 0002 0000 0000 0001 0806 0001  ................
        0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
        0x0020:  0000 0000 0000 0a00 0002                 ..........
18:01:53.409430 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
        0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
        0x0010:  0800 0604 0002 0000 0000 0002 0a00 0002  ................
        0x0020:  0000 0000 0001 0a00 0001                 ..........
18:03:49.367240 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 2145, seq 1, length 64
        0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
        0x0010:  0054 fe72 4000 4001 2834 0a00 0001 0a00  .T.r@.@.(4......
        0x0020:  0002 0800 bb92 0861 0001 f52f c05a 8e7d  .......a.../.Z.}
        0x0030:  0500 0809 0a0b 0c0d 0e0f 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
18:03:49.369478 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 2145, seq 1, length 64
        0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
        0x0010:  0054 478a 0000 4001 1f1d 0a00 0002 0a00  .TG...@.........
        0x0020:  0001 0000 c392 0861 0001 f52f c05a 8e7d  .......a.../.Z.}
        0x0030:  0500 0809 0a0b 0c0d 0e0f 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
```

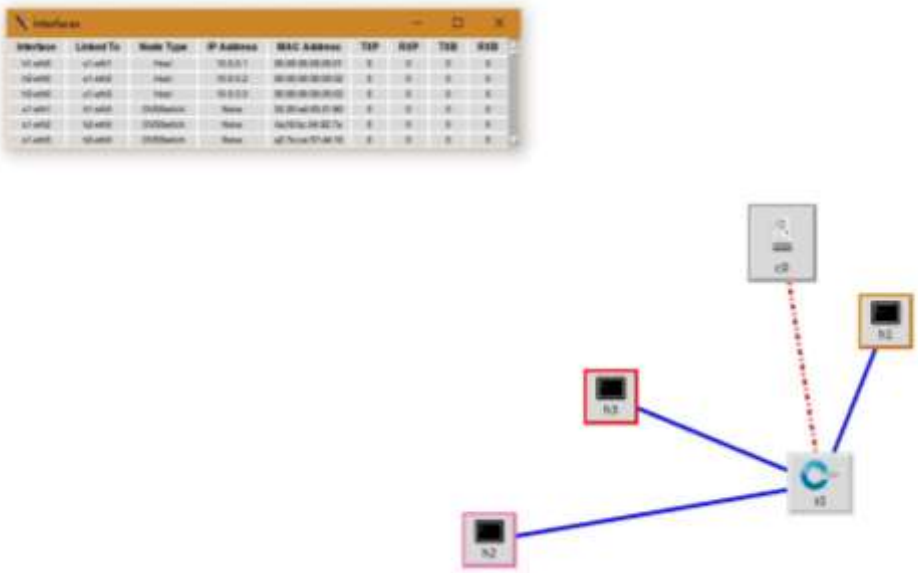
4. Berikan perintah merekam aktivitas node.



5. Tampilan pada miniNAM ketika ping 10.0.1 ke 10.0.2. terjadi komunikasi antar host(gambar 1), sedangkan ketika ping 10.0.1 ke 10.0.5 tidak terjadi komunikasi antar host (gambar 2)



Gambar 1.



Gambar 2.

6. Perbandingan kecepatan menggunakan hub dan switch.

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
mininet> pingpair
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['6.61 Mbits/sec', '7.32 Mbits/sec']
```

7. Menambahkan fungsi act_like_switch ke of_tutorial.py

mininet@mininet-vm: ~/pox/pox/misc

```
def act_like_switch (self, packet, packet_in):
  """
  Implement switch-like behavior.
  """


  # Here's some psuedocode to start you off implementing a learning
  # switch.  You'll need to rewrite it as real Python code.

  # Learn the port for the source MAC
  self.mac_to_port[packet.src] = packet_in.in_port
  if packet.dst in self.mac_to_port:
      print("Packet sent to Control Plane")
    # Send packet out the associated port
    self.resend_packet(packet_in, self.mac_to_port[packet.dst])

    # Once you have the above working, try pushing a flow entry
    # instead of resending the packet (comment out the above and
    # uncomment and complete the below.)

    #log.debug("Installing flow...")
    # Maybe the log statement should have source/destination/port?
    msg = of.ofp_flow_mod()
    msg.match.dl_dst = packet.dst
    ## Set fields to match received packet
    #msg.match = of.ofp_match.from_packet(packet)
    msg.actions.append(of.ofp_action_output(port=self.mac_to_port[packet.dst
    #< Set other fields of flow_mod (timeouts? buffer_id?) >
    self.connection.sed(msg)
    #< Add an output action, and send -- similar to resend_packet() >

  else:
    # Flood the packet out everything but the input port
    # This part looks familiar, right?
    self.resend_packet(packet_in, of.OFPP_ALL)
```