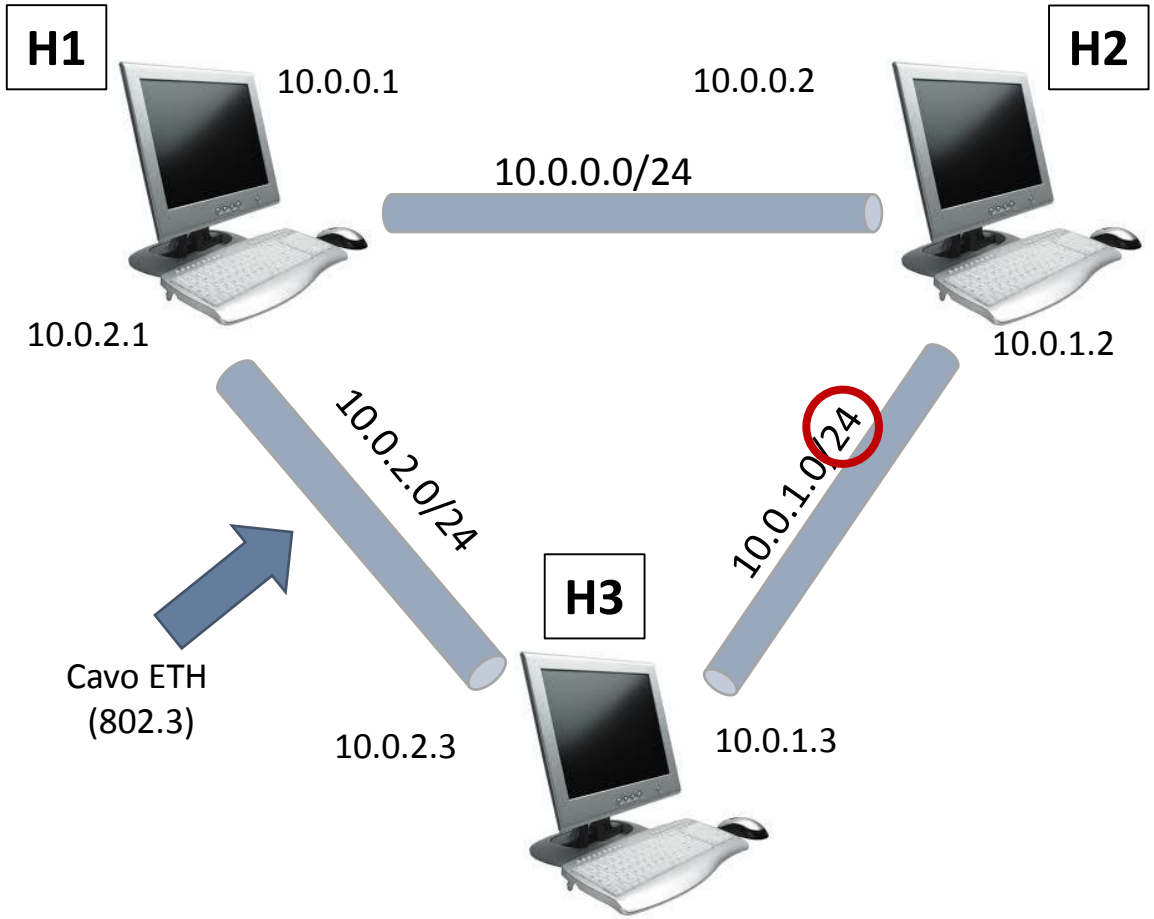
IPv4: Working with routing tables

12/05/2017

Goals

- Understanding the Linux routing tables
- Configure the routing tables of Linux hosts
- Use of ifconfig and route commands

Sample Topology



Implement the topology

- You can modify the file `~/mininet/custom/topo-2sw-2host.py` inside Mininet VM

- To run the experiment on the custom topology, you can use the following command:

```
sudo mn --custom ~/mininet/custom/topo-2sw-2host.py
--topo mytopo –-link tc
```

Topology implementation

```
10
11  from mininet.topo import Topo
12
13  class MyTopo( Topo ):
14      "Simple topology example."
15
16      def __init__( self ):
17          "Create custom topo."
18
19          # Initialize topology
20          Topo.__init__( self )
21
22          # Add hosts and switches
23          Host1 = self.addHost('h1')
24          Host2 = self.addHost('h2')
25          Host3 = self.addHost('h3')
26
27          # Add links
28          self.addLink(Host1, Host2)
29          self.addLink(Host2, Host3)
30          self.addLink(Host1, Host3)
31
32
33  topos = { 'mytopo': ( lambda: MyTopo() ) }
34
```

Configure the NICs using `ifconfig`

```
lsd@sampei:~$ sudo ifconfig
[sudo] password for lsd:
eth0      Link encap:Ethernet  HWaddr 00:50:ba:4b:6c:fe
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 94:de:80:a4:02:8b
          inet addr:151.100.[redacted]  Bcast:151.100.17.255  Mask:255.255.255.0
          inet6 addr: fe80::96de:80ff:fea4:28b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:707516 errors:0 dropped:0 overruns:0 frame:0
          TX packets:191678 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:387133227 (369.1 MiB)  TX bytes:27352871 (26.0 MiB)
          Interrupt:20 Memory:f3300000-f3320000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:9884 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9884 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

Configure the NICs using `ifconfig`

- Disable an interface

```
ifconfig eth0 down
```

- Enable an interface

```
ifconfig eth0 up
```

- Assign IP address to an interface

```
ifconfig eth0 192.168.2.2/24
```

Check the link creation order!

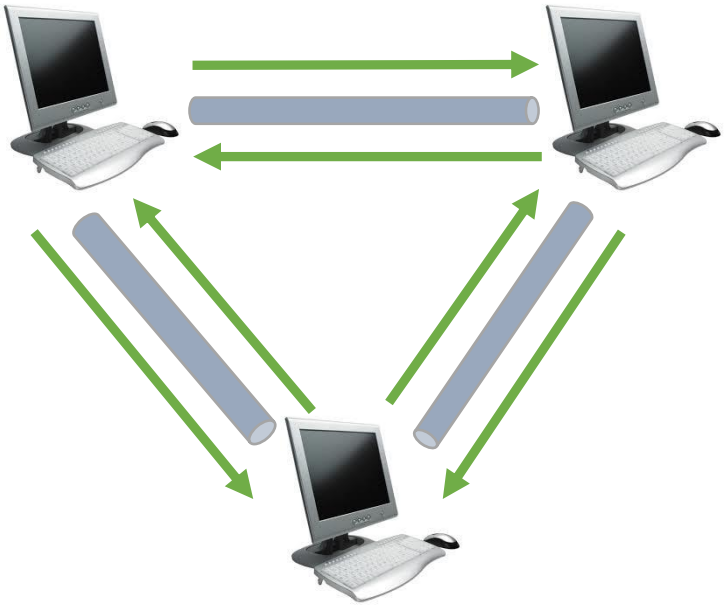
```
mininet@mininet-vm:~$ sudo mn --custom ~/mininet/custom/topo-excercise1.py --topo mytopo --link tc
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:

*** Adding links:
(h1, h2) (h2, h3) (h3, h1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 0 switches
```

- The link creation order affects the ethernet card connections
- The first link is between h1 and h2
  - It means that h1-eth0 is connected to h2-eth0
- The second link is between h2 and h3
  - It means that h2-eth1 is connected to h3-eth0
- And so on…

Configure the NICs using `ifconfig`

- Configure hosts' network interfaces
- Test the connectivity using the `ping` command
- Does everything work fine?

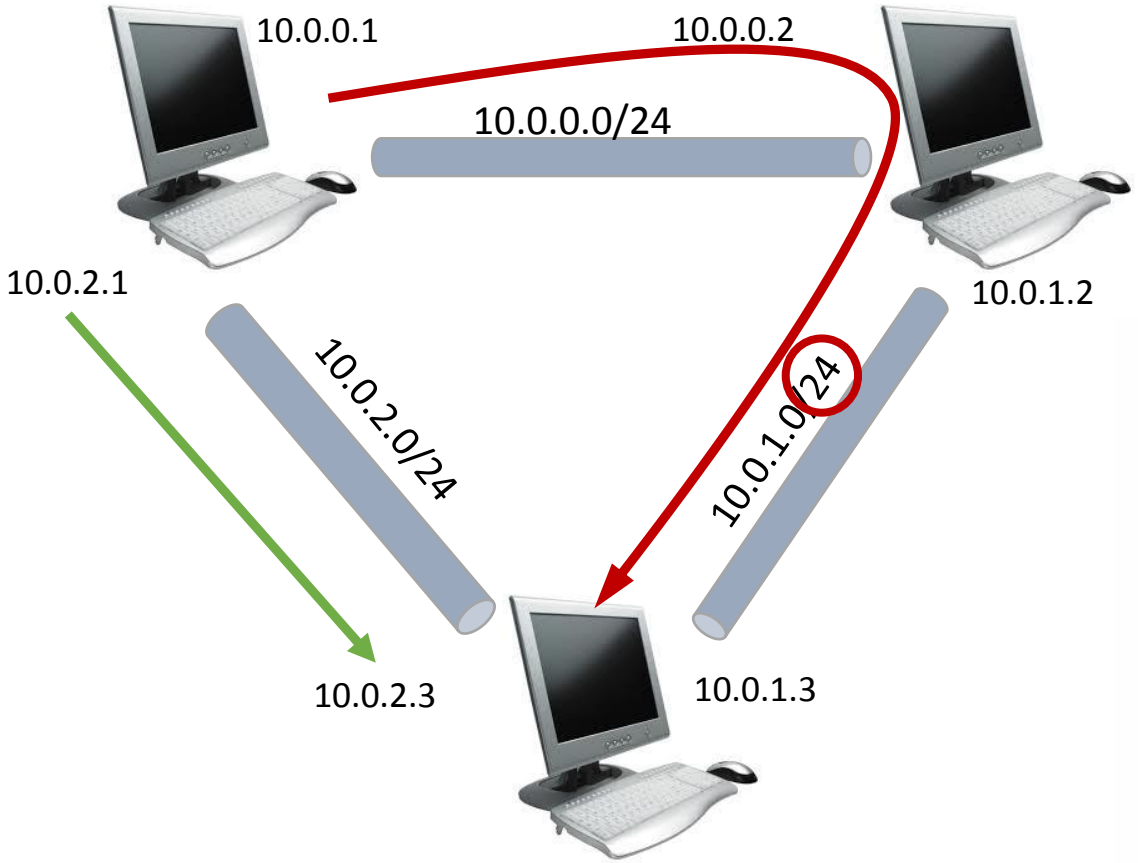


Routing table

- Every host has a routing table
- Lists the routes to other network destinations
- Linux command: `route`

```
lsd@sampei:~$ sudo route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         salaria-gw.di.u 0.0.0.0         UG    1024   0        0 eth1
localnet        *               255.255.255.0   U     0      0        0 eth1
```

Enabling the communication



Update the routing table

- First: update the routing table at both h1 and h3

```
h1:~$ route add –net 10.0.1.0/24 gw 10.0.0.2 dev h1-eth0
h3:~$ route add –net 10.0.0.0/24 gw 10.0.1.2 dev h3-eth0
```

- Second: enabling packet forwarding at h2

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Third: monitor the traffic on h2 using Wireshark!

Increase robustness

- Add one more route from h1 to h3@10.0.1.3

```
h1:~$ route add –net 10.0.1.0/24 gw 10.0.2.3 metric 1 dev
                                    h1-eth1
```

- Question: do we need to add this new rule also on h3?

```
h3:~$ route add –net 10.0.0.0/24 gw 10.0.2.1 dev h3-eth1
```

- Start a ping from h1 to 10.0.1.3 and then disable interface h1-eth0. What's happening?
  - Use Wireshark on h2 to better understand the routing behavior