Brno University of technology

Faculty of Information Technology

Monitoring and Generating Tools of Simple Distance-Vector Protocols

Project documentation

Martin Benes

Contents

1 Introduction **2**
1.1 Task 2
1.2 Motivation 2

2 RIP Protocol **2**
2.1 Messages 2
2.2 Communication 3
2.3 Route Determination 3
2.4 Versions 4
2.5 Problems 5
2.6 Optimalizations 5

3 Implementation **6**
3.1 Sniffer 6
3.2 Fake Responder 7

4 Application **8**
4.1 Testing System Structure 8
4.2 Testing 9

1 Introduction

1.1 Task

The task of the project was to gather informations about RIP protocols, RIPv1, RIPv2 and RIPng and to write two programs in C, one of them RIP sniffer and the other RIP fake responder.

1.2 Motivation

Internet. It is everywhere nowadays - it helps us order pizza, find a taxi, do shopping, connect us with all our friends but it also controls the lights, air condition units etc; it has become a significant part of our lives. There is a lot of data flowing through each its part in every moment. This dataflow is directed (*routed*) by devices called routers, knots in the huge network.

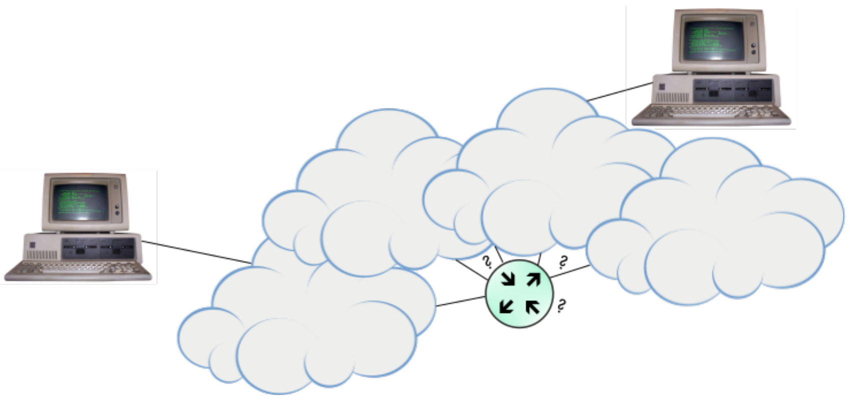


Figure 1: Routing. [1] [2]

Each router has multiple routes connected to it and it has a single task: it recieves a chunk of data (packet) with an address on it and it is supposed to send it closer to the target device in the network as you can see in the figure 1. The router must decide which route should the packet go. And this is exactly what the RIP protocol does.

2 RIP Protocol

Routing Information Protocol is a routing protocol. It enables routers to communicate and to react on the network topology changes. RIP is a distance-vector protocol - no router knows the whole structure of the network. The method it uses to count the shortest path is Bellman-Ford algorithm. A hop count is used as a path cost.

2.1 Messages

The protocol uses UDP transport protocol and it has registered port 520 (RIPv1, RIPv2) and 521 (RIPng). Two basic types of messages are used, RIP Request and RIP Response.

RIP Request RIP Request is sent when the router asks other router for the routing table content (all or just a part).

RIP Response RIP Response includes the routing table of the sender. The destination port is the port in the Source Port array in the RIP Request. But it can be sent even without previous RIP Request received.

2.2 Communication

A newly connected router broadcasts a RIP Request to all interfaces. All the routers running RIP respond sending RIP Response packet containing their routing tables. The router will calculate its own. This process is shown in the figure 2.

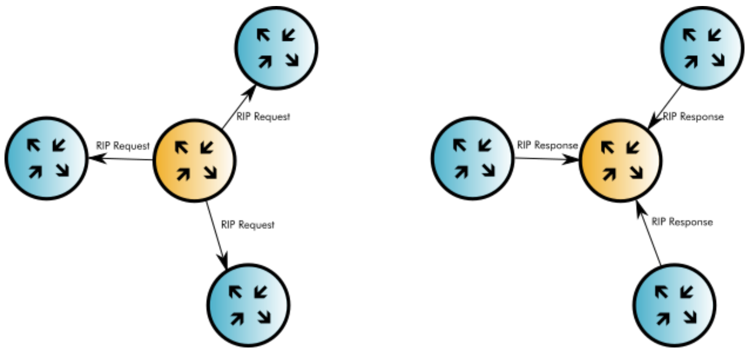


Figure 2: Initialization Process of RIP Protocol.

Then the router can route all the packets coming through. Every route is supposed to regularly broadcast an update, its frequence is set by standard to 30 s. If the update in form of RIP Response comes, the timer will be reset, if not, the route is marked as invalid, the graph is to be seen in the figure 3. After that, "garbage collector timer" 120 s is set for that route and after that the route is removed from the routing table.

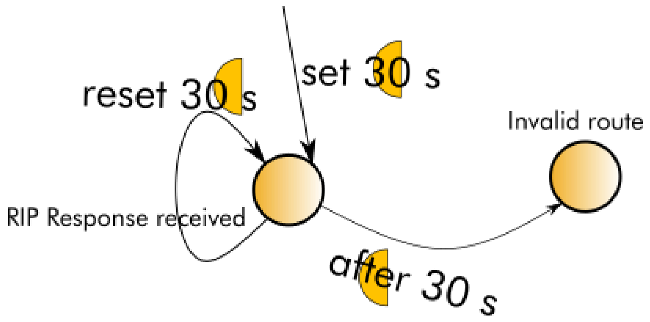


Figure 3: Update Process of RIP Protocol.

If the router detects a change, either an unavailable interface or a new connected neighbor, the triggered update is made. The router broadcast the change using RIP Response packets to all it interfaces. The change is then spread by all routers through the network. This process is described in the figure 4.

2.3 Route Determination

Three main items of each record in the routing table are address of the network/station, distance (hops) and network interface ID. RIP protocol enables routers to share their routing tables and make sending of data through the network much more efficient. The receiving router always compares the new information with the one he has.

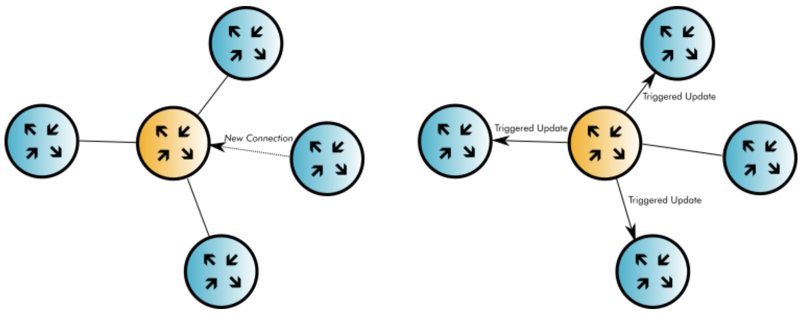


Figure 4: Triggered Update Process of RIP Protocol.

If the network X is reachable in N+1 hops (router increases the number by one), it is compared with the content of the routing table. If the new route is better, it is set and algorithm is propagated to the neighbors to recalculate - this is done again again until the algorithm reaches convergence.

2.4 Versions

RIPv1 Standardized by RFC 1058 (1988) works for routing IPv4 addresses of A, B and C classes. There is no support for network masks (CIDR) - all the networks must have the same mask. A mutual authentication of routers is also not possible, it must be done by different layer. The structure of RIPv1 header and all the packet headers is to be seen in the figures 6 and 6.[3]

RIPv2 The RFC 1388 (1993) introduced the second version of protocol. Some issues were solved, an effort for keeping a backward compatibility caused that RIPv2 let most of the problems of its predecessor stay. The protocol enabled CIDR, as same as the mutual authentication, where MD5 is used. The passwords are sent not encrypted, so it can be very easily attacked. Communication was changed from broadcast to multicast. New route tags provides distinction between routes found by RIP and the routes from different protocols. Headers of the packet with RIPv2 are the same as RIPv1 (figure 5). You can see a RIPv2 header structure in the figure 6. If there is a difference, the RIPv2 is in the parentheses. [4]

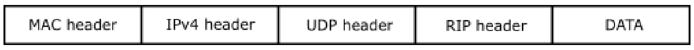


Figure 5: RIPv1 and RIPv2 protocol packet.

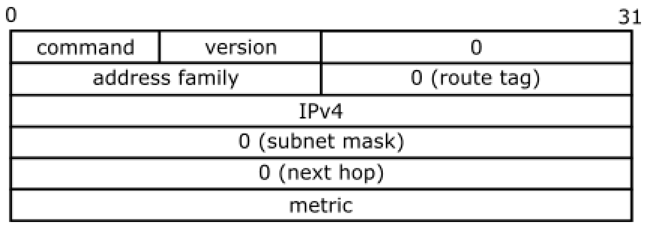


Figure 6: RIPv1 (RIPv2) header. *RIPv2 item is either the same or in parentheses.*

RIPng The RFC 2080 (1997) is a reaction to IPv6. Except of that, it erased support of encryption, that must be covered by IPsec. The structure of packet in the figure 5 stays the same as previous versions, RIPng header is shown in the figure 7. [5] [6]

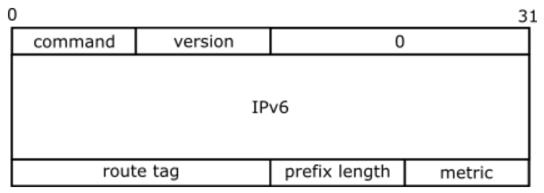


Figure 7: RIPng header.

2.5 Problems

The popularity of the protocol is caused by its simplicity. Although it has some issues which led into replacing RIP protocol in different routing protocols (EIGTP, OSPF).

Slow Convergence Due to the fact that timeout for sending RIP Response is set by RFC to 30 s, convergence of the whole algorithm might take minutes, which is ages for today networing. The propagation of failure is even 180 s, initial delay for propagation. At the networks of hundreds of routers, the time of convergence will become intolerable and the RIP protocol is useless there.

Max hop count The max hop count is limited to 15 - longer paths are not possible, because hop count 16 means unreachable network (distance is infinity). Also, hop count does not always have to be the best metric. It might be better to use delay, load or reliability.

Unsupported CIDR and IPv6 The first version of protocol was not designed for classless addresses and IPv6; The CIDR is possible in RIPv2 and special protocol RIPng was designed for IPv6 support.

2.6 Optimalizations

During each update, RIP floods the network with a lot of traffic. Therefore there are techniques that makes this procedure more efficient.

Split Horizon When the router finds a better route, it is sent to all his neighbors. When Split Horizon is used, the update is not sent to the router that sent us the better route at the first place.

Poison Reverse When the router finds out that some route is unreachable, it immediately updates all its neighbors. [7] [8] [9] [10] [11]

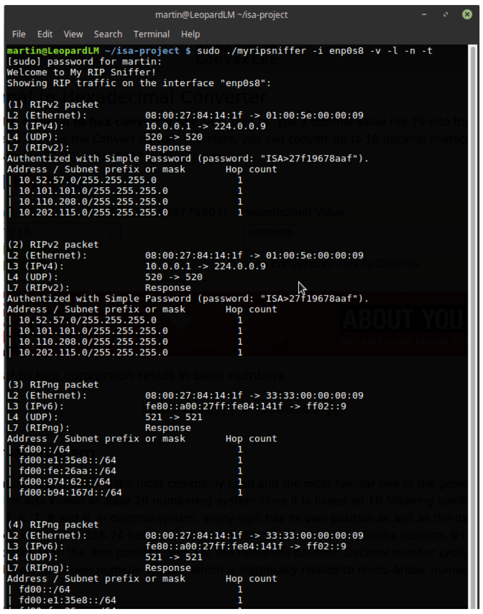
3 Implementation

3.1 Sniffer

First part of project is sniffer, that listens on the given interface and catches RIP communication coming through. It uses *libpcap* library. The usage of the program is:

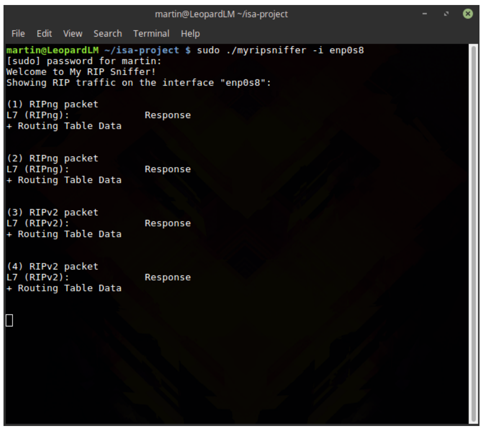
```
sudo ./myripsniffer -i <interface> [-v|--verbose] [-l|--link] [-n|--network] [-t|--transport]
```

The -i parameter is given by task. It specifies the interface program listens to. The -l/-n/-t parameters show additional information in the display about link/network/transport layer. The -v parameter shows full routing informations sent by RIP protocol. You can see full outputs in the figures 9 and 8.



```
martin@LeopardLM ~/isa-project $ sudo ./myripsniffer -i enp0s8 -v -l -n -t
[sudo] password for martin:
Welcome to My RIP Sniffer!
Showing RIP traffic on the interface "enp0s8":

(1) RIPv2 packet
L2 (Ethernet):          08:00:27:84:14:1f -> 01:00:5e:00:00:09
L3 (IPv4):              10.0.0.1 -> 224.0.0.9
L4 (UDP):               520 -> 520
L7 (RIPv2):             Response
Authentized with Simple Password (password: "ISA>27f19678aaf").
Address / Subnet prefix or mask       Hop count
| 10.52.57.0/255.255.255.0              1
| 10.101.101.0/255.255.255.0            1
| 10.110.208.0/255.255.255.0            1
| 10.202.115.0/255.255.255.0            1

(2) RIPv2 packet
L2 (Ethernet):          08:00:27:84:14:1f -> 01:00:5e:00:00:09
L3 (IPv4):              10.0.0.1 -> 224.0.0.9
L4 (UDP):               520 -> 520
L7 (RIPv2):             Response
Authentized with Simple Password (password: "ISA>27f19678aaf").
Address / Subnet prefix or mask       Hop count
| 10.52.57.0/255.255.255.0              1
| 10.101.101.0/255.255.255.0            1
| 10.110.208.0/255.255.255.0            1
| 10.202.115.0/255.255.255.0            1

(3) RIPng packet
L2 (Ethernet):          08:00:27:84:14:1f -> 33:33:00:00:00:09
L3 (IPv6):              fe80::a00:27ff:fe84:141f -> ff02::9
L4 (UDP):               521 -> 521
L7 (RIPng):             Response
Address / Subnet prefix or mask       Hop count
| fd00::/64                             1
| fd00:e1:35e8::/64                     1
| fd00:fe:26aa::/64                     1
| fd00:974:62::/64                      1
| fd00:b94:167d::/64                    1

(4) RIPng packet
L2 (Ethernet):          08:00:27:84:14:1f -> 33:33:00:00:00:09
L3 (IPv6):              fe80::a00:27ff:fe84:141f -> ff02::9
L4 (UDP):               521 -> 521
L7 (RIPng):             Response
Address / Subnet prefix or mask       Hop count
| fd00::/64                             1
| fd00:e1:35e8::/64                     1
```

Figure 8: Verbose output of myripsniffer.



```
martin@LeopardLM ~/isa-project $ sudo ./myripsniffer -i enp0s8
[sudo] password for martin:
Welcome to My RIP Sniffer!
Showing RIP traffic on the interface "enp0s8":

(1) RIPng packet
L7 (RIPng):             Response
+ Routing Table Data

(2) RIPng packet
L7 (RIPng):             Response
+ Routing Table Data

(3) RIPv2 packet
L7 (RIPv2):             Response
+ Routing Table Data

(4) RIPv2 packet
L7 (RIPv2):             Response
+ Routing Table Data
```

Figure 9: Output of myripsniffer.

Whole using of *libpcap* is covered in *sniffer.h*. There is class Sniffer. When method *listen()* is called, the program will start to listen on the interface, until the RIP packet is caught. *Packet* object is returned including all relevant data as strings. The format of output is a matter of caller.

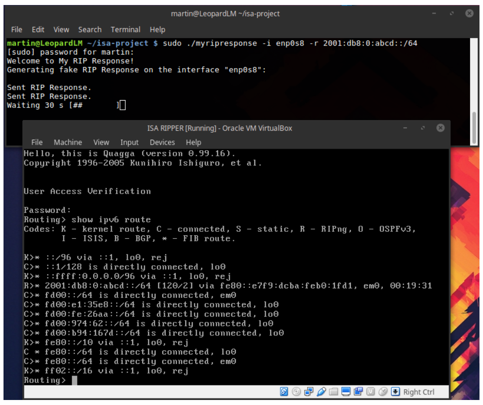
```
do {
    Packet p = s.listen();
    printPacket(p);
} while(true);
```

3.2 Fake Responder

The second part is fake RIP Responder generating traffic to the interface. It causes that routers listening to the link to save the fake route to their routing tables. The usage corresponds with the task specification:

```
sudo ./myripresponse -i <interface> -r <IPv6>/[16-128] {-n <IPv6>} {-m [0-16]} {-t [0-65535]}
```

Parameters -i and -r are compulsory, they specify interface and fake route address. Parameters -n, -m and -t sets respectively next hop, metrics and route tag.


martin@LeopardLM ~/isa-project
File Edit View Search Terminal Help
ISA RIPPER [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
martin@LeopardLM ~/isa-project $ sudo ./myripresponse -i enp0s8 -r 2001:db8:0:abcd::/64
[sudo] password for martin:
Welcome to My RIP Response!
Generating fake RIP Response on the interface "enp0s8":

Sent RIP Response.
Sent RIP Response.
Waiting 30 s [##       ]
```

```
Hello, this is Quagga (version 0.99.16).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password:
Routing> show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng, O - OSPFv3,
       I - ISIS, B - BGP, * - FIB route.

K>* ::/96 via ::1, lo0, rej
C>* ::1/128 is directly connected, lo0
K>* ::ffff:0.0.0.0/96 via ::1, lo0, rej
R>* 2001:db8:0:abcd::/64 [120/2] via fe80::e7f9:dcba:feb0:1fd1, em0, 00:19:31
C>* fd00::/64 is directly connected, em0
C>* fd00:e1:35e8::/64 is directly connected, lo0
C>* fd00:fe:26aa::/64 is directly connected, lo0
C>* fd00:974:62::/64 is directly connected, lo0
C>* fd00:b94:167d::/64 is directly connected, lo0
K>* fe80::/10 via ::1, lo0, rej
C * fe80::/64 is directly connected, lo0
C>* fe80::/64 is directly connected, em0
K>* ff02::/16 via ::1, lo0, rej
Routing>
```

Figure 10: Routes of FreeBSD with the fake route.

After initialization part, where the socket is being opened, program is cycling in a cycle, where it sends packet and waits for 30 s. The interface is to be seen in the figure 10.

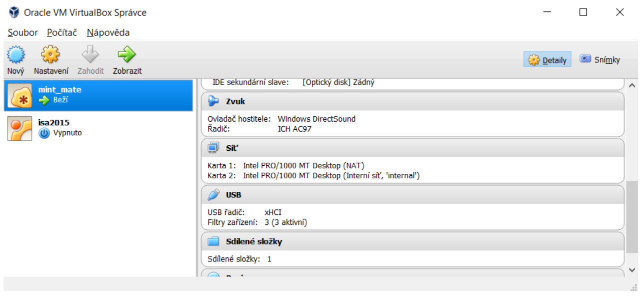
```
do {
    sendto(fd, packet, sizeof(packet), 0, (struct sockaddr*)&addr, sizeof(address));
    sleep(29);
} while(true);
```

4 Application

4.1 Testing System Structure

System where the programs were tested had the structure shown in the figure 14. VirtualBox was used as the virtualization engine. As you can see in the figure 11, virtualized Linux Mint has two interfaces. One of them is set to NAT, linux named it *enp0s3*. The other (*enp0s8*) is host only. It is used for the program to listen to.

Inside the Linux Mint there is another VirtualBox installed, which virtualizes FreeBSD used as virtual router. FreeBSD NIC is bridged to the enp0s8. The figures 12 and 13 display it.



IDE sekundární slave: [Optický disk] Zádný

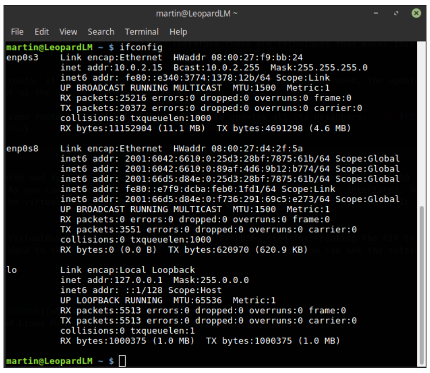
Ovladač hostitele: Windows DirectSound
Řadič: ICH AC97

Karta 1: Intel PRO/1000 MT Desktop (NAT)
Karta 2: Intel PRO/1000 MT Desktop (Interní síť, 'internal')

USB řadič: xHCI
Filtry zařízení: 3 (3 aktivní)

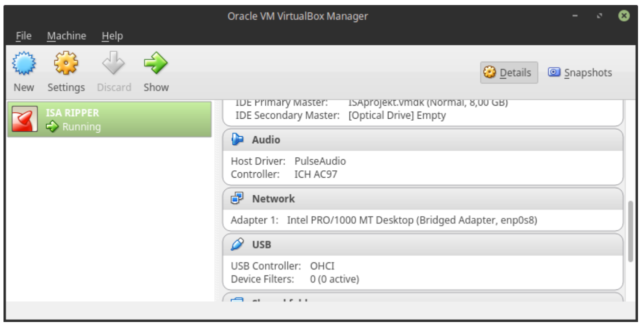
Sdílené složky: 1

Figure 11: Settings of virtualized Linux Mint in VirtualBox on Windows.



```
martin@LeopardLM ~ $ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:f9:bb:24
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::e340:3774:1378:12b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:25216 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20372 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11152904 (11.1 MB)  TX bytes:4691298 (4.6 MB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:d4:2f:5a
          inet6 addr: 2001:6042:6610:0:25d3:28bf:7875:61b/64 Scope:Global
          inet6 addr: 2001:6042:6610:0:89af:4d6:9b12:b774/64 Scope:Global
          inet6 addr: 2001:66d5:d84e:0:25d3:28bf:7875:61b/64 Scope:Global
          inet6 addr: fe80::e7f9:dcba:feb0:1fd1/64 Scope:Link
          inet6 addr: 2001:66d5:d84e:0:f736:291:69c5:e273/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3551 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:620970 (620.9 KB)

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:5513 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5513 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:1000375 (1.0 MB)  TX bytes:1000375 (1.0 MB)

martin@LeopardLM ~ $
```

Figure 12: Interfaces on Linux Mint.



IDE Primary Master: ISAprojekt.vmdk (Normal, 8,00 GB)
IDE Secondary Master: [Optical Drive] Empty

Host Driver: PulseAudio
Controller: ICH AC97

Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, enp0s8)

USB Controller: OHCI
Device Filters: 0 (0 active)

Figure 13: Settings of virtualized FreeBSD in VirtualBox on Linux Mint.

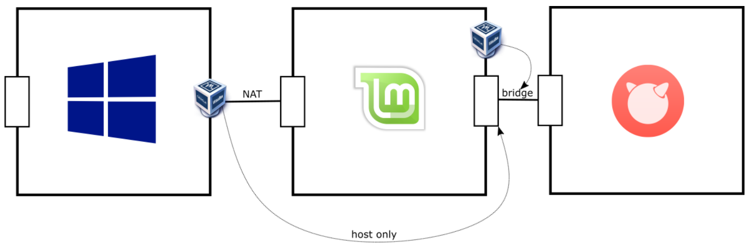


Figure 14: Structure of the whole testing system. [12] [13] [14] [15]

4.2 Testing

Using the sniffer, there were discovered routes for RIPv2 (table 1) and RIPng (table 2).

Address	Mask	Hop Count
10.52.57.0	255.255.255.0	1
10.101.101.0	255.255.255.0	1
10.110.208.0	255.255.255.0	1
10.202.115.0	255.255.255.0	1

Table 1: RIPv2 Routing Table Data.

Address	Subnet Prefix	Hop Count
fd00::	64	1
fd00:e1:35e8::	64	1
fd00:fe:26aa::	64	1
fd00:974:62::	64	1
fd00:b94:167d::	64	1

Table 2: RIPng Routing Table Data.

Testing of responder was verified by checking the routing table in the virtualized FreeBSD. You can see in the figure 10 that traffic generated by the myripresponder caused creating the record in the routing table of the neighbor router.

References

[1] Wikimedia Commons. *Cartoon cloud.* Visited on 2018-10-17. 2009. URL: https://en.m.wikipedia.org/wiki/File:Cartoon_cloud.svg.

[2] Wikimedia Commons. *IBM PC 5150.* Visited on 2018-10-17. 2004. URL: https://en.wikipedia.org/wiki/Personal_computer.

[3] *Routing Information Protocol.* Standard. Rutgers University, June 1988. URL: https://tools.ietf.org/html/rfc1058.

[4] *RIP Version 2.* Standard. Bay Networks, Nov. 1998. URL: https://tools.ietf.org/html/rfc2453.

[5] *RIPng for IPv6.* Standard. Ipsilon Networks, Jan. 1997. URL: https://tools.ietf.org/html/rfc2080.

[6] *RIPng Protocol Applicability Statement.* Standard. Xylogics, Jan. 1997. URL: https://tools.ietf.org/html/rfc2081.

[7] Juniper Networks. *RIP Overview.* Aug. 2017. URL: https://www.juniper.net/documentation/en_US/junos/topics/concept/routing-protocol-rip-security-overview.html.

[8] Charles M. Kozierok. *TCP/IP Routing Information Protocol (RIP, RIP-2 and RIPng).* 2005. URL: http://www.tcpipguide.com/.

[9] *Routing Information Protocol - Wikipedia.* https://en.wikipedia.org/wiki/Routing_Information_Protocol. Accessed: 2018-09-30.

[10] James F. Kurose and Keith W. Ross. *Computer Networking - A Top-Down Approach Featuring the Internet.* Addison Wesley, 2003. ISBN: 0-321-17644-8.

[11] Stephen J. Bigelow. *Mistrovství v počítačových sítích.* Computer Press, 2004. ISBN: 80-251-0178-9.

[12] Wikimedia Commons. *Windows Logo 2012.* Visited on 2018-10-17. 2016. URL: https://commons.wikimedia.org/wiki/File:Windows_logo_%E2%80%93_2012_(blue-purple).svg.

[13] Wikimedia Commons. *Linux Mint logo.* Visited on 2018-10-17. 2014. URL: https://commons.wikimedia.org/wiki/File:Linux_Mint_logo_without_wordmark.svg.

[14] Wikimedia Commons. *FreeBSD Logo.* Visited on 2018-10-17. 2016. URL: https://commons.wikimedia.org/wiki/File:Antu_distributor-logo-freebsd.svg.

[15] Wikimedia Commons. *VirtualBox Logo.* Visited on 2018-10-17. 2010. URL: https://commons.wikimedia.org/wiki/File:Virtualbox_logo.png.