



IP Route 2

Gaurab Raj Upadhaya

Acknowledgement

◆ Linux Advanced Routing and Traffic Control HOWTO

◆ www.lartc.org

Straight to Labwork

- ◆ Iproute2 replaces or supercedes quite a few older linux/unix commands.
- ◆ we'll take a look at those.
- ◆ The functionality is similar so we'll go along and do the lab as the slides proceed

Changes in Lab Configuration

- ◆ Change the NETMASK from existing /21 to /24 in /etc/sysconfig/network-scripts/ifcfg-eth0

- e.g., host1.top.net
 - ◆ IPADDR = 10.1.1.1
 - ◆ NETMASK=255.255.255.0

- ◆ Add a Virtual Interface to your computer

- ◆ create a file /etc/sysconfig/network-scripts/ifcfg-eth0:0 and edit the file. Assign the second IP address from your block to it.
 - e.g host7.bottom.net
 - IPADDR=192.168.7.2
 - NETMASK=255.255.255.0

Changes in Route

- ◆ After changing the NETMASK, you will not be able to ping each other
- ◆ Add a route to the local network
 - `route add -net 192.168.0.0/21 gw 192.168.7.1`
- ◆ To communicate with other computers across your gateway add a default route
 - `route add default gw 10.1.1.1`
- ◆ Test pinging to the second virtual interface on other computer

Now use this commands to see the changes and differences

◆ Start with the traditional 'route' command

◆ route

route

route -n

route -C (to display the routing cache)

route add

route add default

route del

ip route

- ◆ **ip route** provides management tools for manipulating any of the routing tables
- ◆ using the **ip route** is that you can operate on any of the 255 routing tables with this command
- ◆ the **iproute2** suite of tools does not rely on DNS for any operation so, the ubiquitous -n switch in previous examples will not be required in any example
- ◆ **ip route** utility when used in conjunction with the **ip rule** utility can create stateless NAT tables

ip route Operation

- ◆ displaying routes
- ◆ routing cache
- ◆ adding routes
- ◆ deleting routes
- ◆ modifying existing routes
- ◆ fetching a route
- ◆ modifying existing routes

Now use New 'ip' commands

◆ Using ip route to view information

- ip route show
- ip route show cache
- ip route get
- ip route flush
- ip route flush cache

ip route commands

◆ adding a route

- ip route add x.x.x.x/nn via x.x.x.x
 - ◆ e.g, ip route add 192.168.0.0/21 via 192.168.7.1

◆ adding a default route

- ip route add default via x.x.x.x
 - ◆ eg. ip route add default via 10.1.1.1

◆ IP Route Add Prohibit

- used to prohibit routing to specified destination
 - ◆ ip route add prohibit x.x.x.x from x.x.x.x
 - ◆ eg. (in host1) ip route add prohibit 172.16.1.1 from 192.168.0.0/21

◆ IP Route NAT

- easier to implement NAT with ip route
 - ◆ ip route add nat x.x.x.x via n.n.n.n, where x.x.x.x= destination, n.n.n.n = interface
 - ◆ e.g, (in host1) ip route add nat 0.0.0.0 via 172.16.1.2

ip route show

- ◆ use the ip route show command to see changes
- ◆ also use 'ip route get' to see specific routes
- ◆ after applying a route change, flush the route cache with ip route flush cache to see the changes

ip rule 2

- ◆ It is another part of the **iproute2** software package
- ◆ **ip rule** is the single tool for manipulating the routing policy database under linux (RPDB)
- ◆ IP RULE let you deploy policy based routing in your linux router
- ◆ IP Rules lets you modify local routing tables as well as source and destination based routing

ip rule commands

- ◆ ip rule show

- ◆ ip rule add

- add a route to the routing table

- ◆ ip route add 10.1.0.0/21 via 10.1.1.1 table 6

- ◆ ip rule add tos 0x08 table 6

- ◆ ip rule show

- ◆ ip rule syntax are similar to other ip commands

Traffic Control

◆ using ip route to do traffic control

◆ What can be done

- Throttle bandwidth TO certain computers
- Help you to fairly share your bandwidth
- Protect your network from DoS attacks
- Protect the Internet from your customers
- Multiplex several servers as one, for load balancing or enhanced availability
- Restrict access to your computers
- Limit access of your users to other hosts
- Do routing based on user id (yes!), MAC address, source IP address, port, type of service, time of day, or content

iproute2

◆ Linux Kernel / IPROUTE

- Linux has a sophisticated system for bandwidth provisioning called Traffic Control.
- This system supports various method for classifying, prioritizing, sharing, and limiting both inbound and outbound traffic.
- the netfilter package is now integrated in the Kernel to provide these advanced features

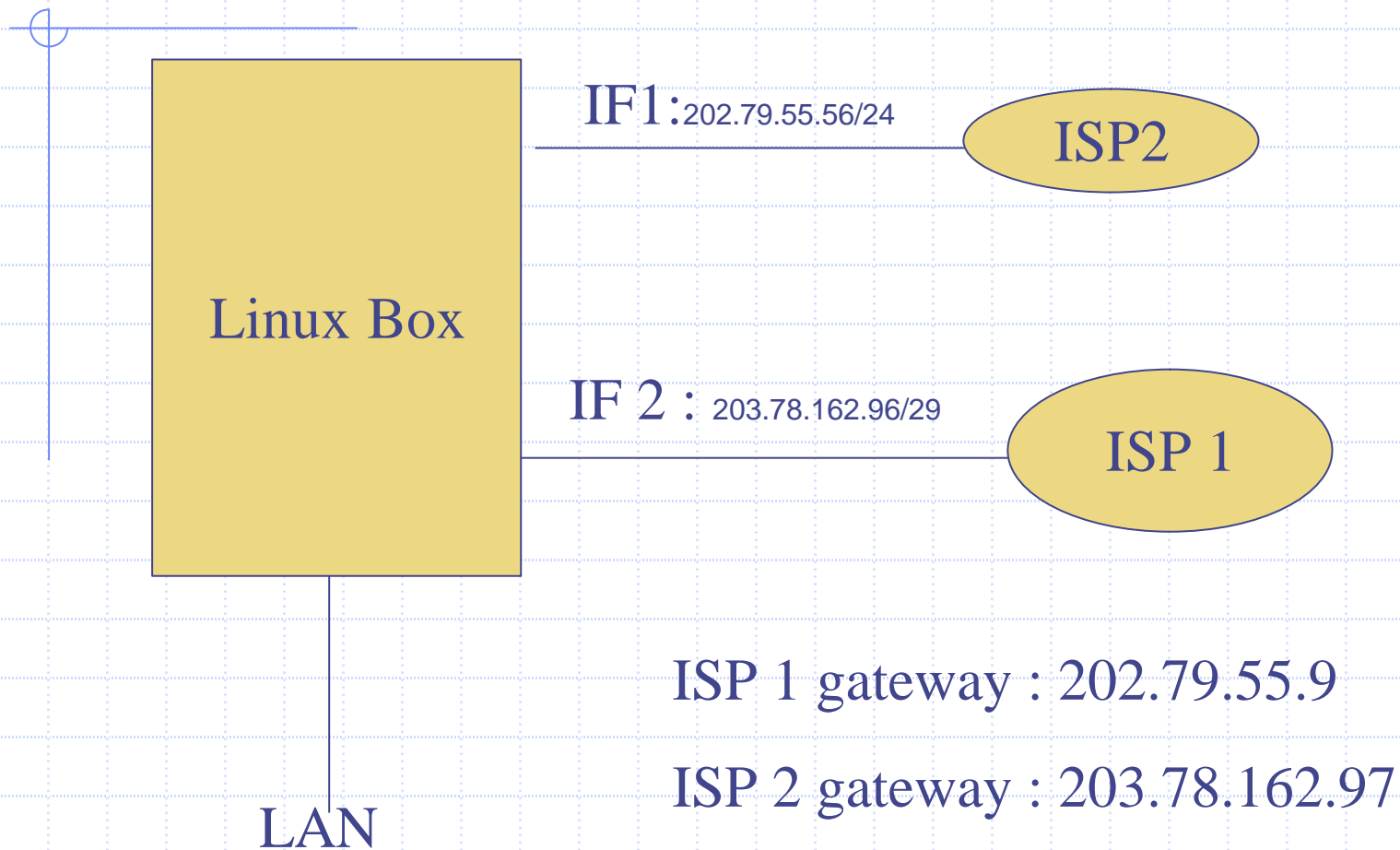
◆ Requirements

- ◆ Any Linux distribution such as RedHat or Debian
- ◆ Iproute2 package

source policy routing

```
[root@classroom root]# ip route show table local
nat 0.0.0.0 via 203.78.162.100 scope host
broadcast 192.168.100.0 dev eth1 proto kernel scope link src 192.168.100.100
local 203.78.162.100 dev eth0 proto kernel scope host src 203.78.162.100
broadcast 127.255.255.255 dev lo proto kernel scope link src 127.0.0.1
broadcast 203.78.162.103 dev eth0 proto kernel scope link src 203.78.162.100
broadcast 203.78.162.96 dev eth0 proto kernel scope link src 203.78.162.100
local 192.168.100.100 dev eth1 proto kernel scope host src 192.168.100.100
broadcast 192.168.100.255 dev eth1 proto kernel scope link src 192.168.100.100
broadcast 127.0.0.0 dev lo proto kernel scope link src 127.0.0.1
local 127.0.0.1 dev lo proto kernel scope host src 127.0.0.1
local 127.0.0.0/8 dev lo proto kernel scope host src 127.0.0.1
[root@classroom root]#
```


Routing for multiple providers



IP route Configurations

- ◆ to route answers to packets coming in over a particular provider, say ISP 1, back out again over that same ISP creates two additional routing tables, say ISP1 and ISP2.

- echo 1 ISP1 >> /etc/iproute2/rt_tables
- echo 1 ISP2 >> /etc/iproute2/rt_tables

Then you set up routing in these tables as follows:

```
ip route add 202.79.55.0 dev eth1 src 202.79.55.56 table ISP1
ip route add default via 202.79.55.9 table ISP1
ip route add 203.78.162.96 dev eth2 src 203.78.162.100 table
ISP2
ip route add default via 203.78.162.97 table ISP2
```

set up the main routing table.

```
ip route add 202.79.55.0 dev eth1 src 202.79.55.56
ip route add 203.78.162.96 dev eth2 src 203.78.162.100
```

preference for default route:

```
_ip route add default via 203.78.162.97
```

set up the routing rules.

```
ip rule add from 202.79.55.0 table ISP1
```

```
ip rule add from 203.78.162.100 table ISP2
```

Load balancing

◆ Instead of choosing one of the two providers as your default route, you now set up the default route to be a multipath route. In the default kernel this will balance routes over the two providers.

- `ip route add default scope global nexthop via 203.78.162.97 dev eth2 weight 1`
- `nexthop via 202.79.55.9 dev eth1 weight 1`

◆ balancing will not be perfect, as it is route based, and routes are cached. This means that routes to often-used sites will always be over the same provider.



Queuing Disciplines for Bandwidth Management

Queues and Queueing

- ◆ queueing we determine the way in which data is *SENT*.
- ◆ we can only shape data that we transmit.

classless Queueing

- ◆ Classless queueing disciplines are those that, by and large accept data and only reschedule, delay or drop it.
- ◆ These can be used to shape traffic for an entire interface, without any subdivisions.

Queuing mechanisms

- ◆ FIFO, First In First Out
- ◆ Packets arrive and leave the queue in exactly the same order
- ◆ Simple configuration and fast operation
- ◆ No Priority servicing or bandwidth guarantees possible
- ◆ WFQ, Weighted Fair Queuing
- ◆ A hashing algorithm, places flows into separate queues where weights are used to determine how many packets are serviced at a time. You define weights by setting IP Precedence and DSCP values.
- ◆ Simple configuration.
- ◆ No priority servicing or bandwidth guarantees possible.

Queuing mechanisms (2)

- ◆ CQ, Custom Queuing

- ◆ Traffic is classified into multiple queues with configurable queue limits.
- ◆ Has been available for a few years and allows approximate bandwidth allocation for different queues.
- ◆ No priority servicing possible. Bandwidth guarantees are approximate and there are a limited number of queues. Configuration is relatively difficult.

- ◆ PQ, Priority Queuing

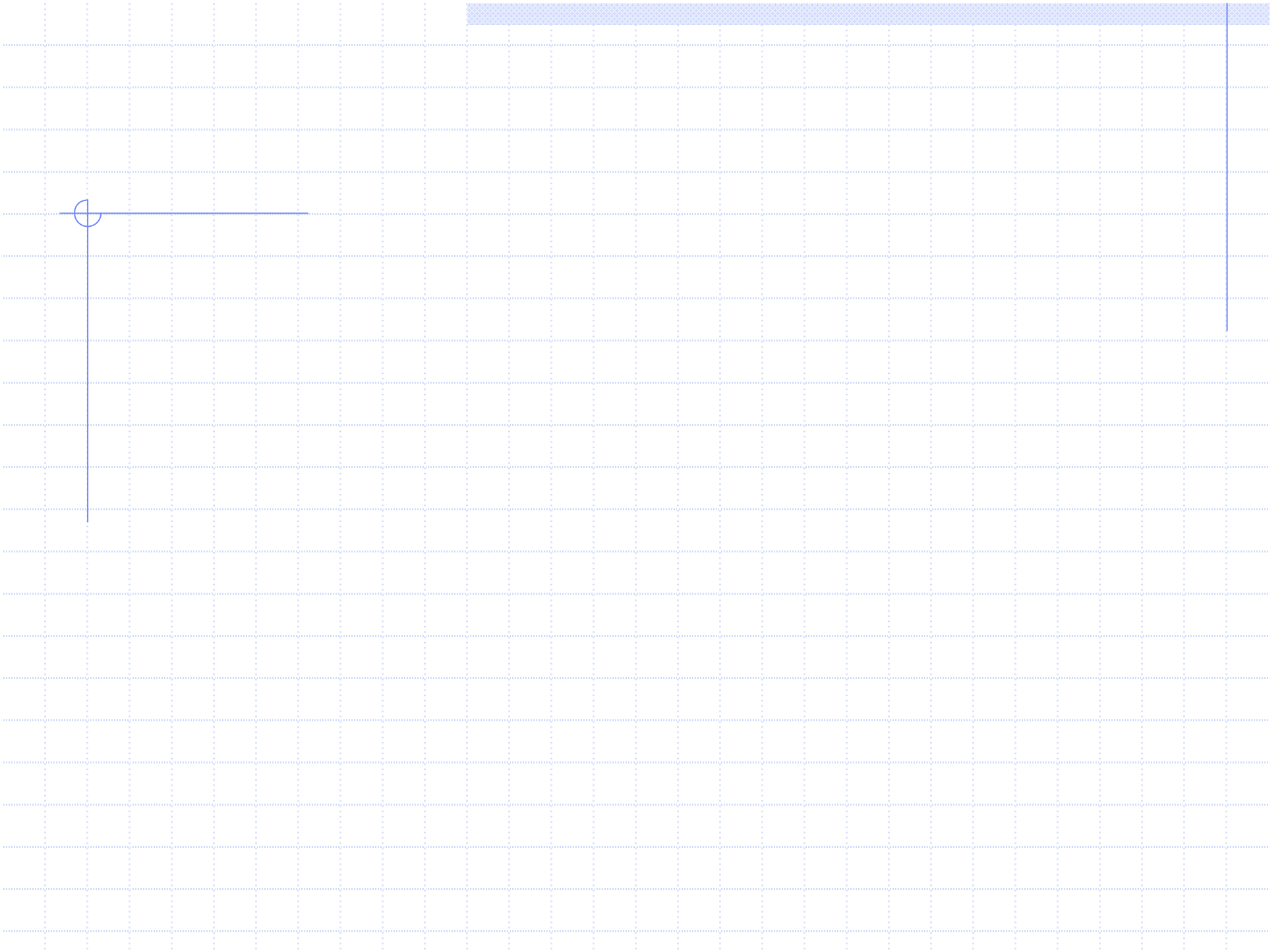
- ◆ Traffic is classified into high, medium, normal and low priority traffic is serviced first, then medium priority traffic, followed by normal and low priority traffic.
- ◆ Has been available for a few years and provides priority servicing.
- ◆ Higher priority traffic can starve lower priority queues of bandwidth. No bandwidth guarantees possible.

Queuing mechanisms (3)

- ◆ CBWFQ, Class Based Weighted Fair Queuing
- ◆ MQC is used to classify traffic. Classified traffic is placed into reserved bandwidth queues or a default unreserved queue.
- ◆ Similar to LLQ except there is no priority queue. Simple configuration and ability to provide bandwidth guarantees. No priority servicing possible.
- ◆ PQ-WFQ, Priority queue-Weighted Fair Queuing (IP RTP Priority)
- ◆ Single interface command is used to provide priority servicing to all UDP packets destined to even port numbers within a specific range.
- ◆ Simple, one command config. Provides priority servicing to RTP packets.
- ◆ All other traffic is treated with WFQ. RTCP traffic is not prioritized. No guaranteed bandwidth capability.
- ◆ Note: MQC = Modular QoS CLI

Queuing mechanisms (4)

- ◆ Low Latency Queueing (LLQ) = Priority Queue (PQ) + Class Based-Weighted Fair Queue (CB-WFQ).
- ◆ Allows a strict Priority Queue to handle a defined class of packet to be prioritized over all other traffic.
- ◆ Simple config, ability to provide priority to multiple classes of traffic and give upper bounds on priority bandwidth utilization. Can also config bandwidth guaranteed classes and a default class.
- ◆ All priority traffic is sent through the same priority queue which can introduce jitter.





Dynamic Routing with Zebra

Routing Protocols

◆ Interior Routing Protocols

- RIP , IGRP , OSPF

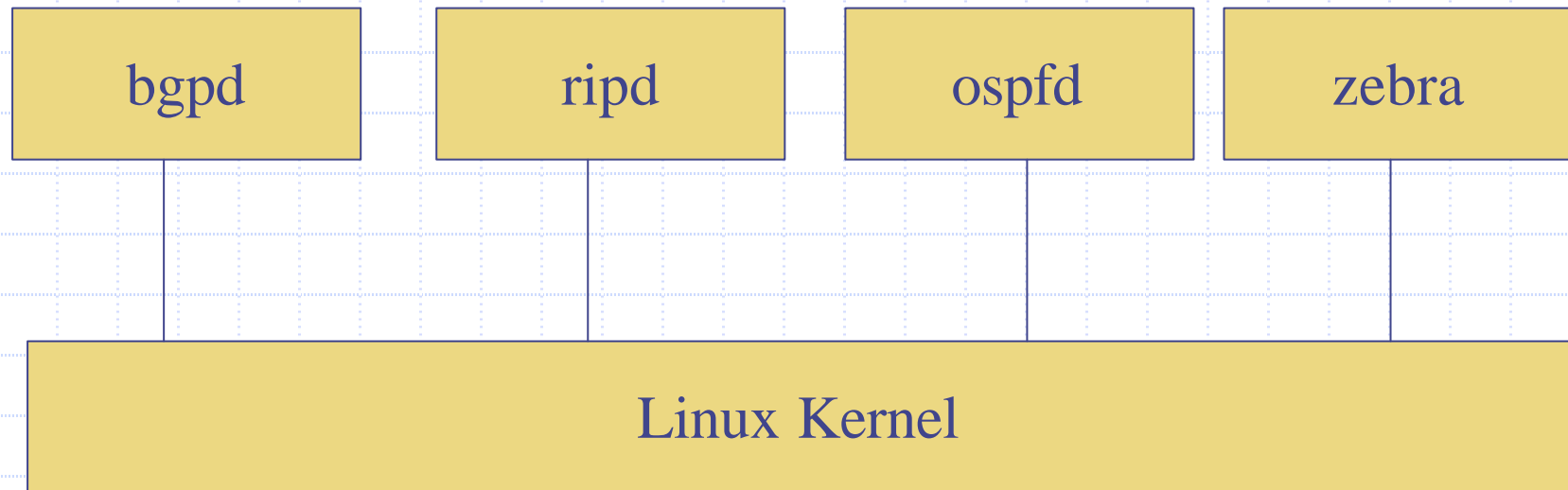
◆ Exterior Routing Protocols

- BGP

Zebra

- ◆ GNU Zebra is free software that manages TCP/IP based routing protocols.
- ◆ It supports BGP-4 protocol as described in RFC1771 (A Border Gateway Protocol 4) as well as RIPv1, RIPv2 and OSPFv2
- ◆ Zebra is unique in its design in that it has a process for each protocol
- ◆ Zebra also supports special BGP Route Reflector and Route Server behavior

System Architecture



Supported Platforms

- ◆ GNU/Linux 2.0.37
- ◆ GNU/Linux 2.2.x
- ◆ GNU/Linux 2.3.x
- ◆ FreeBSD 2.2.8
- ◆ FreeBSD 3.x
- ◆ FreeBSD 4.x
- ◆ NetBSD 1.4
- ◆ OpenBSD 2.5
- ◆ Solaris 2.6
- ◆ Solaris 7

Basic commands

- ◆ There are five routing daemons in use, and there is one manager daemon
 - ripd, ripngd, ospfd, ospf6d, bgpd
 - zebra

Installing

- ◆ Get the zebra rpm package which comes with the common distributions
- ◆ Install zebra on your machine
- ◆ Make sure that there is zebra.conf file in /etc
- ◆ If not there then create one

Running Zebra

```
[root@classroom root]# service zebra start
```

```
Starting zebra:
```

```
[ OK ]
```

```
[root@classroom root]#
```

Zebra.conf file

```
[root@classroom root]#  
[root@classroom root]#  
[root@classroom root]# clear  
[root@classroom root]# service zebra start  
Starting zebra:  
[root@classroom root]# vi /etc/zebra.conf  
hostname classroom  
password kcm  
enable password cisco  
line vty  
login  
password kishor
```

[OK]

```
[root@classroom root]# telnet localhost zebra
```

```
Trying 127.0.0.1...
```

```
Connected to localhost.
```

```
Escape character is '^]'.
```

```
Hello, this is zebra (version 0.93b).
```

```
Copyright 1996-2002 Kunihiro Ishiguro.
```

```
User Access Verification
```

```
Password:
```

```
classroom> en
```

```
Password:
```

```
classroom#
```

```
classroom#
```

```
classroom# show ip route
```

```
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,  
        B - BGP, > - selected route, * - FIB route
```

```
K>* 0.0.0.0/0 via 203.78.162.97, eth0
```

```
K>* 10.38.0.0/16 via 192.168.100.100, eth1
```

```
K * 127.0.0.0/8 is directly connected, lo
```

```
C>* 127.0.0.0/8 is directly connected, lo
```

```
K>* 169.254.0.0/16 is directly connected, eth1
```

```
K>* 172.16.100.0/24 via 192.168.100.1, eth1
```

```
K>* 192.168.10.0/24 via 192.168.100.1, eth1
```

```
[root@classroom root]# telnet localhost zebra
```

```
Trying 127.0.0.1...
```

```
Connected to localhost.
```

```
Escape character is '^]'.
```

```
Hello, this is zebra (version 0.93b).
```

```
Copyright 1996-2002 Kunihiro Ishiguro.
```

```
User Access Verification
```

```
Password:
```

```
classroom> en
```

```
Password:
```

```
classroom#
```

```
classroom# config t
```

```
classroom(config)#
```

```
classroom(config)#
```

```
classroom(config)# int
```

```
IFNAME Interface's name
```

```
classroom(config)# int eth1
```

```
classroom(config-if)#
```

```
classroom# show int eth1
```

```
Interface eth1
```

```
index 3 metric 1 mtu 1500 <UP,BROADCAST,RUNNING,MULTICAST>
```

```
HWaddr: 00:80:ad:02:63:ce
```

```
inet 192.168.100.100/24 broadcast 192.168.100.255
```

```
input packets 6372092, bytes 1165025360, dropped 0, multicast packets 0
```


```
input errors 0, length 0, overrun 0, CRC 0, frame 0, fifo 0, missed 0
```

```
output packets 4943561, bytes 2147483647, dropped 0
```

```
output errors 0, aborted 0, carrier 0, fifo 0, heartbeat 0, window 0
```

```
collisions 336139
```

```
classroom#
```

```
[root@classroom root]# service ospfd start
```

```
Starting ospfd:
```

```
[ OK ]
```

```
[root@classroom root]# service bgpd start
```

```
[root@classroom root]#
```

```
[root@classroom root]# service ospfd start
```

```
Starting ospfd:
```

```
[ OK ]
```

```
[root@classroom root]# service bgpd start
```

```
[root@classroom root]# telnet localhost ospfd
```

```
Trying 127.0.0.1...
```

```
Connected to localhost.
```

```
Escape character is '^]'.
```

```
Hello, this is zebra (version 0.93b).
```

```
Copyright 1996-2002 Kunihiro Ishiguro.
```

```
User Access Verification
```

```
Password:
```

```
kishor>
```

```
kishor>
```

```
kishor>
```

```
kishor> en
```

```
Password:
```

```
Password:
```

```
kishor#
```

```
kishor#
```

```
kishor#
```

```
kishor#
```

```
kishor# show ip route
```

```
% Unknown command.
```

```
kishor# show ip ospf
```

```
OSPF Routing Process not enabled
```

```
kishor#
```

kishor(config-router)#

area	OSPF area parameters
auto-cost	Calculate OSPF interface cost according to bandwidth
capability	Enable specific OSPF feature
compatible	OSPF compatibility list
default-information	Control distribution of default information
default-metric	Set metric of redistributed routes
distance	Define an administrative distance
distribute-list	Filter networks in routing updates
end	End current mode and change to enable mode.
exit	Exit current mode and down to previous mode
help	Description of the interactive help system
list	Print command list
mpls-te	Configure MPLS-TE parameters
neighbor	Specify neighbor router
network	Enable routing on an IP network
no	Negate a command or set its defaults
ospf	OSPF specific commands
passive-interface	Suppress routing updates on an interface
quit	Exit current mode and down to previous mode
redistribute	Redistribute information from another routing protocol
refresh	Adjust refresh parameters
router-id	router-id for the OSPF process
timers	Adjust routing timers
write	Write running configuration to memory, network, or terminal

```
kishor(config-router)# network  
  A.B.C.D/M  OSPF network prefix  
kishor(config-router)# network 192.168.100.0  
% There is no matched command.  
kishor(config-router)# network 192.168.100.0  
% Unknown command.  
kishor(config-router)# network 192.168.100.0/24  
% Command incomplete.  
kishor(config-router)# network 192.168.100.0/24  
  area  Set the OSPF area ID  
kishor(config-router)# network 192.168.100.0/24 area  
  <0-4294967295>  OSPF area ID as a decimal value
```