

CHAPTER 4

DYNAMIC ROUTING

In this chapter we cover the basic principles of dynamic routing algorithms that form the basis of the experiments in Lab 4.

The chapter has six sections. Each section covers material that you need to run the lab exercises. The first section gives an overview of dynamic routing protocols and discusses the differences between the two major classes of routing algorithms: intra domain and inter domain. Sections 2, 3, 4, and 5 give an overview of the most common routing algorithms such as RIP, OSPF, BGP and IGRP. Section 6 presents the commands used to configure the hosts and the routers for dynamic routing.

TABLE OF CONTENT

1	ROUTING PROTOCOLS	3
1.1	AUTONOMOUS SYSTEMS (AS)	3
1.2	INTRADOMAIN ROUTING VERSUS INTERDOMAIN ROUTING	4
1.3	DYNAMIC ROUTING	4
1.3.1	<i>Distance Vector Algorithm</i>	4
2	RIP	5
3	OSPF	5
4	BGP	5
5	TOOLS AND UTILITIES	5
5.1	CONFIGURING DYNAMIC ROUTING ON A CISCO ROUTER	5
5.2	CONFIGURING DYNAMIC ROUTING ON A LINUX PC.....	8

1 Routing protocols

Routing protocols consist of communication primitives that are used explicitly for the exchange of routing information. The exchanged information is then used by a routing algorithm to determine the *optimal* path to a destination and the creation of a routing table. Many routing algorithms exist. They all differ in the information that is used to calculate the path from a source to a destination (e.g., local versus global), the metrics and thresholds for selection, the frequency of updates, the rate of convergence, etc. In this chapter we will discuss some of the more common algorithms that have been implemented for route calculation between hosts and routers within a domain and between just routers for inter domain routing.

1.1 Autonomous systems (AS)

An autonomous system (AS) is an administrative entity that controls and administers a network of hosts and routers within a single domain. The network control and operation inside an AS is local and not subject to any external supervision. The gateway or router that interconnects the AS to another AS or ASs, must comply to the standard protocols and procedures that are used to communicate between such domains. An AS must provide a consistent picture of its addressing space to the external world, i.e., which hosts/IP addresses are reachable via its gateways.

ASs came into being when the size of the Internet became too unwieldy. As more and more routers were added to the Internet, it became obvious that updating and maintaining a single large network of routers was very difficult, the Internet lost its flexibility and ability to adapt to changes. In the early 80's it was decided to carve up the Internet into administrative domains, that would control their own network devices and use a *gateway*¹ to connect their network, called a *stub*, to a *core* network that would serve as the backbone to connect to other stub networks (domains). Each AS is allocated a number by the IANA. These numbers are 16 bit numbers that can be either public (in the case of an ISP providing transit services) or private (a campus/corporate network).

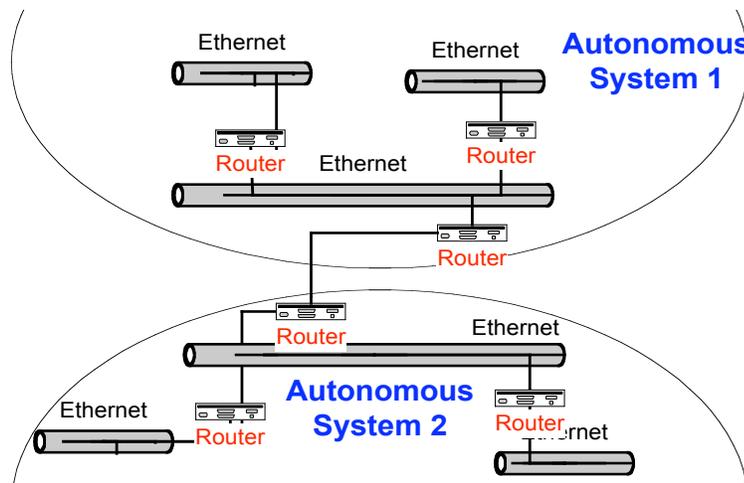


Figure 1. Autonomous Systems

¹ Thus the use of the name gateway to refer to a router that connects two domains.

1.2 Intradomain routing versus Interdomain routing

Intradomain routing consists of the exchange of route information between devices that are within a single AS whereas interdomain routing consists of the exchange of reachability information between devices that lie on the boundary of ASs. As such, interdomain routing is often referred to as a reachability protocol more so than a dynamic routing algorithm.

Intradomain routing need not follow a standard or use a single routing algorithm, its internal operation is hidden from the rest of the world, it can pick and chose which algorithms to run on its subnets. On the other hand, interdomain routing needs to follow the guidelines set forth by the Internet body so that no miscommunication can occur that could result in the termination of communication services between two domains. A domain is generally a much smaller entity and easier to administer than the global Internet. The routing protocols can therefore consist of more complex and dynamic schemes that exploit node and link state information. The interdomain routing protocols are less adaptive, they need to converge faster and must be robust to sudden changes in network state. Although no dynamic algorithm is loop free (therefore the TTL field in the IP header), recovery from transients should be fast in interdomain routing algorithms. This could come at the cost of less adaptability.

1.3 Dynamic routing

The role of a router and a host are very different when it comes to routing as explained in Chapter 3. A host is only responsible for maintaining a routing table, it is a silent/passive observer of the routing process. A host listens to routing messages that are broadcast on the local subnet (e.g. router advertisements) to fill in the necessary entries in its routing tables. A router on the other hand, is responsible for:

- Identifying neighbors (other routers)
- Discovering routes (from other routers)
- Selecting a route (next hop to a destination)
- Maintaining routing information (dynamically participating in router message exchanges)

Depending on where a router falls in the AS, it may be responsible for only interior routing (e.g., a router inside an AS), only exterior routing (e.g., a router in a backbone WAN) or maybe both (e.g., a gateway router, one interface is interior whereas the other is exterior).

The most common dynamic routing protocols use very different algorithms for path determination. The distance vector based routing algorithms exchange information only with its immediate neighbors and uses that to make hop by hop routing decisions, the link state based algorithm exchanges full state information with *all* the nodes in a specified area, the path vector.

1.3.1 Distance Vector Algorithm

- 2 RIP
- 3 OSPF
- 4 BGP
- 5 Tools and Utilities

5.1 Configuring dynamic routing on a Cisco router

Cisco routers support most routing protocols, including BGP, RIP, and OSPF, and can be configured through the Command Line Interface (CLI). **Figure 2** is a brief overview of the IOS CLI hierarchy, indicating what commands can be entered in which mode. Similar to the configuring router interfaces, routing protocols can be configured through the Router Configuration Mode (config-router).

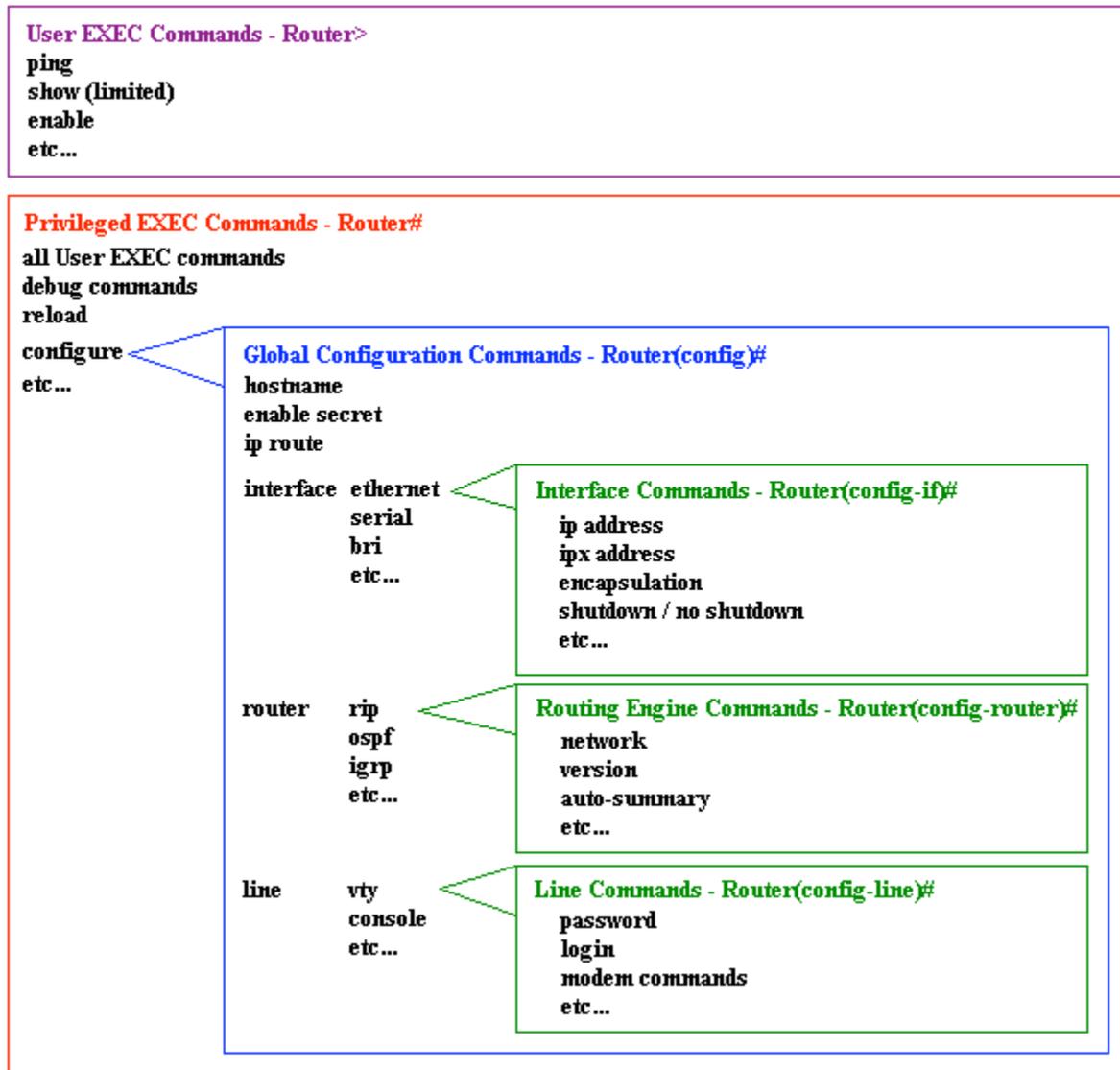


Figure 2: IOS CLI Hierarchy

The following table summarizes the commonly used commands to enable RIP on a Cisco router.

Router1(config)# router rip Router1(config-router)#	Enables the RIP routing process. This will take you to the router configuration mode.
Router1(config)# no router rip	Disables the RIP routing process.
Router1(config-router)# network ip_netaddr	Associates network <i>ip_netaddr</i> to RIP. RIP sends updates ONLY to the interfaces with this network address. <i>ip_netaddr</i> shall not include the subnet id. More than one network can be added.
Router1(config-router)# no network ip_netaddr	Disables RIP for the specified network.
Router1(config-router)# passive-interface interface	Sets the specified <i>interface</i> to RIP passive mode.
Router1(config-router)# no passive-interface interface	Sets the specified <i>interface</i> to RIP non-passive mode.
Router1(config-router)# offset-list 0 in value interface	Increases all the incoming metrics in RIP packets received at specified <i>interface</i> by <i>value</i> .
Router1(config-router)# offset-list 0 out value interface	Increases all the outgoing metrics in RIP packets sent from specified <i>interface</i> by <i>value</i> .
Router1(config-router)# no offset-list 0 in/out value interface	Disables the specified offset-list.
Router1(config-router)# version 2	Sets the RIP version to 2.
Router1(config-router)# timers basic update invalid hold-down flush <i>Example:</i> Router1(config-router)# timers basic 30 180 180 240	Sets the following RIP timers: <i>update</i> – the rate at which routing updates are sent. <i>invalid</i> – the interval after which a route is declared invalid. <i>hold-down</i> – the interval during which routing information regarding better paths is suppressed. <i>flush</i> – the amount of time that must pass before a route is removed from the routing table
Router1(config-router)# flash-update-	If the update is due in <i>time</i> seconds then do

threshold time	not send a triggered update. Thus, if <i>time</i> is set to a value equal to the <i>update timer</i> then triggered updates are disabled.
-----------------------	---

Table 1: Cisco IOS commands for RIP.

The following table summarizes the commonly used commands to enable OSPF on a Cisco router.

Router1(config)# router ospf process-id Router1(config-router)#	Enables OSPF routing process. This will take you to the router configuration mode. (Next Line) <i>process-id</i> is a numeric value local to the router. This enables one router to have multiple OSPF processes. However, in this lab, you will only use one process. Assign 1 for the <i>process-id</i> . Note that it does not have to match <i>process_ids</i> on other routers.
Router1(config)# no router ospf process-id	Disables the specified OSPF process.
Router1(config-router)# network ip_netaddr wildcard_mask area area_id Example: Router1(config-router)# network 10.0.0.0 0.255.255.255 area 1	<i>ip_addr</i> is the network address on which the OSPF process runs. <i>Wildcard_mask</i> helps in reducing configuration lines. 0 is a match bit and 1 is a don't care bit. The <i>area_id</i> is the area that interfaces are in specified by the <i>ip_addr</i> and <i>wildcard_mask</i> . (From the example, all interfaces whose 1 st byte equals 10 belongs to area 1). <i>Area 0</i> is reserved for Backbone area.
Router1(config-router)# no network ip_addr wildcard_mask area area_id	Disables OSPF on the specified network area.
Router1(config-router)# passive-interface interface	Sets the specified interface (eth0) to passive mode.
Router1(config-router) no passive-interface interface	Sets the specified interface (eth0) to non-passive mode (actively participates in OSFP algorithm).
Router1# show ip ospf	Displays general information about OSPF
Router1# show ip ospf database	Displays the link-state database
Router1# show ip ospf border-routers	Displays the Area Border Router (ABR) and Autonomous System Boundary Router (ASBR).
Router1# clear ip ospf process-id process	Resets the specified OSPF process.

Table 2: Cisco IOS commands for OSPF.

The following table is a review of several commands available to obtain the configuration statistics of a Cisco router.

Router1# show ip route	Displays the entries in the routing table.
Router1# clear ip route *	Deletes all the entries from the routing table, except the ones associated with its own interfaces.
Router1# show ip cache	Displays the entries in the routing cache.
Router1# clear arp-cache	Deletes all the entries in the ARP table, except the ones associated with its own interfaces.
Router1(config)# ip routing	Enables IP routing.
Router1(config)# no ip routing	Disables IP routing. This also resets the all the configurations related to routing.

Table 3: Cisco IOS commands for viewing statistics.

5.2 Configuring dynamic routing on a Linux PC

Zebra is a configuration software for managing TCP/IP based routing protocols, including RIP, OSPF, and BGP, on a Linux PC. Zebra uses multithread technology under multithread supported UNIX kernels, and allows for a separate process to be run for each protocol. Each module (routing daemon) of zebra can be started, stopped, configured, and upgraded independently of the others, providing protection from the failure of one protocol affecting the entire system. Supported routing protocols are:

<i>bgpd</i>	Manages BGP-4 and BGP-4+ protocol
<i>ripd</i>	Manages RIPv1, v2 protocol
<i>ripngd</i>	Manages RIPng protocol
<i>ospfd</i>	Manages OSPFv2 protocol
<i>ospf6d</i>	Manages OSPFv3 protocol

Zebra acts as a kernel routing table manager and is responsible for changing the kernel routing table and for redistribution of routes between different routing protocols, as shown in the diagram below.

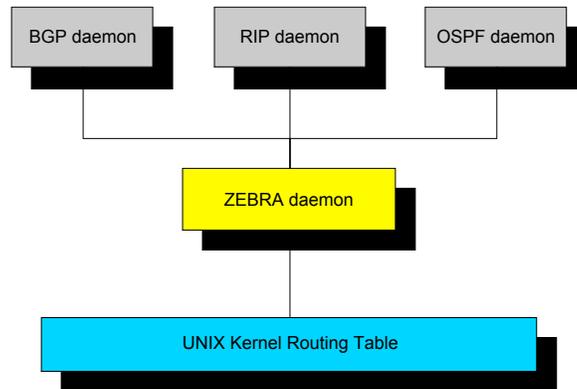


Figure 3: Zebra System Architecture

Thus, in order to properly update the kernel routing table, the zebra daemon must be running prior to starting and configuring the routing protocols. Zebra daemons have their own configuration file which defines the default settings of each protocol at system boot-up, and are generally found in the directory `/usr/local/etc/` or `/etc/zebra/`. They are basically empty since most of the features can be configured dynamically just like the Cisco routers once the daemon are running. In addition, zebra daemons have their own terminal interface or VTY, which can be connected via a telnet session to the appropriate port number, and is defined by the file `/etc/services`.

```

zebrasrv      2600/tcp      # zebra service
zebra         2601/tcp      # zebra vty
ripd          2602/tcp      # RIPd vty
ripngd        2603/tcp      # RIPngd vty
ospfd         2604/tcp      # OSPFd vty
bgpd          2605/tcp      # BGPd vty
ospf6d        2606/tcp      # OSPF6d vty
  
```

On Red Hat 7.3 Linux systems, shell scripts located in `/etc/rc.d/init.d/` are used to start and stop the zebra daemons. Since these scripts can be lengthy to type, aliases have been created for the lab exercises to make the procedure easier.

<u>Alias</u>	<u>Command</u>
<code>runzebra</code>	<code>/etc/rc.d/init.d/zebra</code>
<code>runbgpd</code>	<code>/etc/rc.d/init.d/bgpd</code>
<code>runospfd</code>	<code>/etc/rc.d/init.d/ospfd</code>
<code>runripd</code>	<code>/etc/rc.d/init.d/ripd</code>

Arguments accepted by the above commands are:

```
start | stop | restart | reload | condrestart | status
```

You may check whether the daemons are already running by using the `status` option:

```
PC1% runzebra status
```

If there is more than one process of each daemon running, terminate the processes by either using the `stop` option:

```
PC1% runzebra stop
```

or with the `pkill` command:

```
PC1% pkill zebra
```

For example, to set up a routing process, first start the `zebra` daemon, and then the routing protocol daemon, such as `ripd`:

```
PC1% /etc/rc.d/init.d/zebra start
PC1% /etc/rc.d/init.d/ripd start
```

Now connect to and configure the `ripd` daemon:

```
PC1% telnet localhost 2602
ripd> enable
ripd# configure terminal
ripd(config)# router rip
ripd(config-router)# version 2
ripd(config-router)# network 10.0.0.0/8
ripd(config-router)# passive-interface eth0
ripd(config-router)# redistribute connected
ripd(config-router)# end
ripd# show ip rip
ripd# exit
```

As one can see, the interface and commands for the `ripd` daemon are similar to those used on a Cisco router. This is also the case for configuring other protocols within zebra.

References:

<http://www.zebra.org>

<http://www.cisco.com/warp/cpropub/45/tutorial.htm>

http://www.cisco.com/univercd/cc/td/doc/product/software/ios112/112cg_cr/5cbook/5ciprout.htm