

CSIS0234B Computer and Communication Networks (Class B)

Reading for Tutorial 9

Dynamic Routing with Zebra

In the previous tutorials, we see that it is possible to manually create routing tables in Linux to suit any network, and make sure that every pair of hosts will communicate with the optimal route. However, the strategy is practical only if the network is small and never change in topology. In a dynamic environment like the Internet, the tremendous amount of effort required makes it completely infeasible. Instead, the routers has to compute the routing table by themselves. Within an autonomous system of the Internet, an “Interior Gateway Protocol” (IGP) is used; among autonomous systems an “Exterior Gateway Protocol” (EGP) is used instead¹. Commonly used routing algorithms includes the Distance Vector (also called Bellman-Ford) algorithm and the Link-State Routing algorithm. Protocols that implement such algorithms are widely in use. For Distance vector, there is RIP (Routing Information Protocol) version 1 and 2 and RIPng (the ng stands for “New Generation”, which supports IPv6). For Link State algorithms there is OSPF (Open Shortest Path First) protocol version 2 (for IPv4) and 3 (for IPv6). All these are IGPs; for EGP there is only one protocol in wide-spread use, the BGP (“Border Gateway Protocol”) version 4 which is based on distance vector. We will focus on the OSPF algorithm. Since it uses link state routing, it is a good idea to review the textbook or notes about how such algorithm runs.

Routing algorithms are never hard-coded into an OS kernel. Instead a user-level daemon is executed in order to send and receive information from the peer, and hence calculate the preferred routing table. The routing table is then installed into the kernel. There are many routing daemon written for Unix, including `routed` and `gated`. One of the most versatile routing daemon ever written is the GNU `zebra`, which implements all the routing protocols mentioned in the previous paragraph. The downside is that documentation is very thin, as it does not yet reach the 1.0 release (current release is 0.93b). We will have some experience on how dynamic routing behaves by configuring and looking into the operation of `zebra`.

1. OSPF

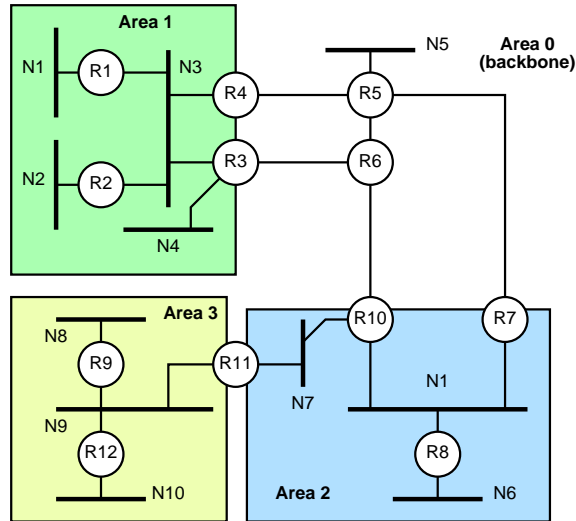
OSPF extends the basic link state routing algorithm in the following ways:

1. It allows a network with both broadcast and point-to-point media (rather than just point-to-point links). This is done by abstracting the network using a graph that contains only point-to-point links. See the textbook figure 5-64, or lecture notes p7.23.
2. In a broadcast network, all messages required by the OSPF algorithm is delivered to all OSPF routers at the same time using multicasts. A separate multicast address (224.0.0.5) is allocated to mean “all OSPF routers within the same LAN”. To avoid sending the same information about the network many times, only one of the routers in a network, called its *designated* router, will broadcast link-state information of the network itself.
3. It allows external routing information to be distributed within the network. In particular, each network can have some **AS boundary** routers that talks to routers in other ASes using an EGP. The AS boundary routers thus compute the path to other ASes, and send distribute tables of routing costs throughout the whole network. Each router within the AS can thus

¹The most important difference between IGP and EGP is that while in IGP the primary concern is cost of routing, for EGP the primary concern is to find just any path (without a cycle) that the system administrator doesn’t hate. It is because each AS may uses a completely different metric, so there is no way to compare costs among ASes.

find a path to all other ASes connected.

4. It allows the adjacent sub-network to form **areas**, so that the link state algorithm is only executed within the area to reduce routing overhead. Each sub-network is in exactly one area. A router can have connections to multiple areas (an *area-border router*), or it can have connection to only one of them (an *internal router*). Each area has a 32-bit identification. The area 0 is called the **backbone area**, in the sense that all inter-area traffic will go to the backbone once. Here is an example of such partitioning:



Note that every area should be completely connected, except the backbone area which may be separated by areas. All routers connected to multiple areas must be backbone routers, e.g., R11 is a backbone router even it has no direct connection without going through an area. The backbone area may use an area as a “virtual link” (i.e., tunnel) so that all backbone routers can communicate (perhaps indirectly). E.g., A virtual link may be configured between R11 and R7, and between R11 and R7, through area 2. This allows the backbone area to run the link-state algorithm.

An area-border router runs multiple copies of the link-state algorithm, one per area. It is then responsible for distributing the information to all backbone router. Upon reception of such information, it must then find the cost from itself to all other routers (whether or not it is in the same area), and distribute the result to all areas that it connects to. This allows, for example, a host in N1 to send to a host in N10 by first sending it to the best exit of area 1 (R3), go through the backbone to the best entry of area 3 (R11, by going through R6, R10, and then R11 through the virtual link), and then send to N10 through the route within area 3.

2. Zebra

Zebra is actually a suite of routing daemon rather than a single daemon. Administrators can select a set of routing protocols to support, and runs only the subset of routing daemons. E.g., to run only the OSPF protocol, one only need to run the `ospfd` daemon. Such daemons do not install the routes found by themselves. Instead, the information are passed to a central daemon (called `zebra`), which will combine the information obtained from different routing protocols and install the routes.

2.1. Configuring Zebra

Before one setup the daemons themselves, one must stop all interfaces. `zebra` is very reluctant to turn an otherwise working network to unusable, that it refuse to install the routes found by various daemons if the interfaces and routing tables are installed before it is executed. The complete network setup must be done by `zebra`. Luckily, the setup is not very complicated.

All these daemons are configured in the same way. One should start them (using “`/etc/init.d/zebra start`”, “`/etc/init.d/ospfd start`”, etc) after the configuration is done. The configuration files are stored in `/etc/zebra/xxx.conf`, where the `xxx` is `zebra`, `ospfd`, etc. Once the service is running you can control the service and see the information about it by connecting to a TCP port for the daemon, i.e., `telnet localhost xxx` (where `xxx` can be `zebra`, `ospfd`, etc). If you’d like to shut down the server at a later time, just replace `start` word of the command above by the `stop` word. If you restart the `zebra` server, you should also restart all the actual routing servers.

The main `zebra.conf` configuration file is responsible for setting up the hostnames, interfaces and IP addresses of the router. All configuration files can also setup the password to use the (telnet-based) control interface. As an example,

```
hostname router01
password zebra
enable password secret
interface lo
    ip address 127.0.0.1/8
    no shutdown
interface eth0
    ip address 192.168.0.3/24
    multicast
    no shutdown
ip route 0.0.0.0/0 192.168.0.1
```

There are two passwords, the first one for the control interface (i.e., connected using “telnet”), and the second one for enabling commands which modifies the behaviour of the router. After that is the interface setup, and you will have one per interface available. Each will be assigned an address (including the number of bits for the network part). Then a “no shutdown” command will ensure the interface is up (in most cases, prefixing “no” in a command would undo the command). A “multicast” command enable multicasting over the network, which should always be done whenever you want to use OSPF (because it uses multicasting itself).

The last line shows how to add a “static route” for the router, i.e., a route that does not come from a routing daemon. It is possible to ask `zebra` to “redistribute” static routes of a router to the OSPF algorithm, as we will see later. However, a default route (one with 0-length network part like the above) is never redistributed. If desired you can break it into two parts so that such static routes can be distributed.

2.2. Configuring OSPF daemon

The `ospfd` daemon is configured in `ospfd.conf`. As in `zebra.conf`, you can use it to setup the passwords of the control interface. You should also use it to enable OSPF routing (by default it is disabled even after you execute the daemon, to be enabled in the control interface), and to tell which of the networks should OSPF be used, which must also specify what area your router

belongs to. As an example,

```
password zebra
router ospf                ! enable OSPF routing
redistribute static        ! ask zebra to redistribute routes
network 160.80.40.20/26 area 0 ! use OSPF on the interface for this subnet,
                               ! which is in the backbone area
```

Note that for two routers in the same network to communicate routing information, they must both have the interface to that network setup for OSPF routing.

The configuration file can modify many other parameters, including the “metric” of a network (i.e., cost to go through a network, by default 10), how network delay can be scaled to automatically compute a metric, the amount of time between HELLO and link state broadcast, authentication, etc. For details you can refer to the manual, which is available from the course web page. Exactly the same commands can also be executed in the control interface of the OSPF daemon, to further configure `ospfd` when it is already running.

2.3. Showing OSPF daemon interface

Once OSPF is up and running, you can examine the current status of the router by using the control interface. Some of the more interesting commands are listed here:

- `show ip ospf`: show basic information about the router.
- `show ip ospf interface`: show information about the all or a particular interface.
- `show ip ospf neighbor`: show information about the all or a particular neighbour.
- `show ip ospf database`: show the link state database stored.
- `show ip ospf route`: show the routes installed by OSPF.