Chapter 12
Open Shortest Path First (OSPF)

12.1 OSPF Overview

Open Shortest Path First Routing (OSPF) is a shortest path first or link-state protocol. OSPF is an interior gateway protocol that distributes routing information between routers in a single autonomous system (AS). OSPF chooses the least-cost path as the best path. OSPF is suitable for complex networks with a large number of routers because it provides equal-cost multi-path routing, where packets to a single destination can be sent via more than one interface simultaneously.

In a link-state protocol, each router maintains a database describing the entire AS topology, which it builds out of the collected link state advertisements of all routers. Each participating router distributes its local state (i.e., the router's usable interfaces and reachable neighbors) throughout the AS by flooding. Each multi-access network that has at least two attached routers has a designated router and a backup designated router. The designated router floods a link state advertisement for the multi-access network and has other special responsibilities. The designated router concept reduces the number of adjacencies required on a multi-access network.

OSPF allows networks to be grouped into areas. Routing information passed between areas is abstracted, potentially allowing a significant reduction in routing traffic. OSPF uses the following four different types of routes, listed in order of preference: intra-area, inter-area, type 1 Autonomous System External (ASE), and type 2 ASE. Intra-area paths have destinations within the same area. Inter-area paths have destinations in other OSPF areas. Both types of ASE routes are routes to destinations external to OSPF (and usually external to the AS). Routes exported into OSPF ASE as type 1 ASE routes (see "Exporting to OSPF ASE and NSSA" on page 151) are supposed to be from interior gateway protocols (such as RIP) whose external metrics are directly comparable to OSPF metrics. When a routing decision is being made, OSPF will add the internal cost to the AS border router to the external metric. Type 2 ASEs are used for exterior gateway protocols whose metrics are not comparable to OSPF metrics. In this case, only the internal OSPF cost to the AS border router is used in the routing decision.

From the topology database, each router constructs a tree of the shortest paths, with itself as the root. This shortest-path tree gives the route to each destination in the AS. Externally-derived routing information appears on the tree as leaves. The link-state advertisement format distinguishes between information acquired from external sources and information acquired from internal routers, so there is no ambiguity about the source or reliability of routes. Externally-derived routing information (for example, routes learned from BGP) is passed transparently through the AS and is kept separate from OSPF's internally derived

data. Each external route can also be tagged by the advertising router, enabling routers on the borders of the AS to pass additional information between them.

OSPF optionally includes Type of Service (TOS) routing and allows administrators to install multiple routes to a given destination for each type of services, such as low delay or high throughput. A router running OSPF uses the destination address and the type of service to choose the best route to the destination.

OSPF intra- and inter-area routes are always imported into the GateD routing database with a preference of 10. Because it would violate the protocol if an OSPF router did not participate fully in the area's OSPF, it is not possible to override this preference. Although it is possible to give other routes better preference values explicitly, doing so would violate the OSPF protocol and could lead to incorrect routing. Therefore, you cannot specify import or export policy for OSPF; you can only specify export policy for OSPF ASE.

Hardware multicast capabilities are also used where possible to deliver link-status messages. OSPF areas are connected by the backbone area, the area with identifier 0.0.0.0. All areas must be logically contiguous, and the backbone is no exception. To permit maximum flexibility, OSPF allows the configuration of virtual links, which enables the backbone area to appear contiguous despite the physical reality.

Because a separate copy of the link-state algorithm is run for each area, most configuration parameters are defined on a per-area basis. All routers in an area must agree on that area's parameters. Misconfiguration will keep neighbors from forming adjacencies between themselves, and routing information might not flow or could loop.

GateD can run over a variety of physical connections: serial connections, LAN interfaces, ATM, or FDDI. The OSPF configuration supports three different types of connections in the interface clauses:

LAN and Point-to-Point

An example of a LAN interface is an Ethernet or a FDDI interface. A point-to-point interface can be a serial line using Point-to-Point protocol. GateD will use a Multicast IP address on LAN interfaces to reach OSPF routers.

Non-Broadcast Multiple Access

ATM with virtual circuits is an example of a Non-Broadcast Multiple Access medium. Because there is no general multicast in all ATM devices, each router must be listed so that GateD can poll each router. GateD will unicast the packets to the routers in the NBMA network.

Point-to-Multipoint

Point-to-Multipoint connectivity is used when the network does not provide full connectivity to all routers in the network. Just as on the NBMA format, you must provide a list of routers that the GateD deamon will query as OSPF peers.

12.2 OSPF Syntax

```
ospf on | off [ {
    always-update-summary on | off ;
    retransmitinterval global_default_time ;
```

```
transitdelay global_default_time ;
priority global_default_priority ;
hellointerval global_time ;
routerdeadinterval global_default_time ;
pollinterval global_default_time ;
advertise-subnet on | off ;
opaque-capability on | off ;
auth [none | simple auth_key | md5 md5-keyset]  ;
defaults {
    preference defasepref ;
    cost defasecost ;
    tag [ as ] tagvalue ;
    type 1 | 2 ;
    inherit-metric ;
    ribs unicast [ multicast ] ;
    nssa-preference defnssapref ;
    nssa-cost defnssacost ;
    nssa-type 1 | 2 ;
    nssa-inherit-metric ;
};
traceoptions trace_options_ospf ;
rfc1583compatibility on | off ;
area areanumber | backbone {
    nssa [ cost defaultcost type 1 | 2 ] ;
    nssanetworks {
        network mask stubmask [ restrict ] ;
        network masklen number [ restrict ] ;
        host stubhost [ restrict ] ;
    };
    stub [ cost stub_default_cost] ;
    stubhosts {
        host cost cost ;
    };
    stubnetworks {
        network mask stubmask cost cost ;
        network masklen number cost cost ;
        host stubhost cost cost ;
    };
```

```
networks {
    network [ restrict ] ;
    network mask netmask [ restrict ] ;
    network masklen number [ restrict ] ;
    host nethost [ restrict ] ;
} ;
summaryfilters {
    route_filter
} ;
retransmitinterval area_default_time ;
transitdelay area_default_time ;
priority area_default_priority ;
hellointerval area_time ;
routerdeadinterval area_default_time ;
pollinterval area_default_time ;
advertise-subnet on | off ;
auth [none | simple auth_key | md5 md5-keyset]  ;
interface interface_list
    [ cost ifcost ]
    [ { enable | disable ;
    retransmitinterval iftime ;
    transitdelay iftime ;
    priority ifpriority ;
    hellointerval if_time ;
    routerdeadinterval iftime ;
    pollinterval iftime ;
    passive ;
    advertise-subnet on | off ;
    auth [none | simple auth_key | md5 md5-keyset] ;
    } ] ;
interface interface_name | interface_address nonbroadcast
    [ cost ifnbcost ]
    [ { strict-routers on | off ;
    routers {
        gatewaylist [ eligible ] ;
    } ;
    retransmitinterval ifnbtime ;
    transitdelay ifnbtime ;
```

```
            priority ifnbpriority ;
            hellointerval ifnb_time ;
            routerdeadinterval ifnbtime ;
            pollinterval ifnbtime ;
            passive ;
            advertise-subnet on | off ;
            auth [none | simple auth_key | md5 md5-keyset] ;
            } ] ;
        interface interface_name | interface_address point-to-multipoint
            [ cost ptmcost ]
            [ { strict-routers on | off ;
            routers {
                gatewaylist ;
            } ;
            retransmitinterval ptmtime ;
            transitdelay ptmtime ;
            priority ptmpriority ;
            hellointerval ptmtime ;
            routerdeadinterval ptmtime ;
            pollinterval ptmtime ;
            passive ;
            advertise-subnet on | off ;
            auth [none | simple auth_key | md5 md5-keyset]  ;
            } ] ;
```

Backbone only:

```
virtuallink neighborid router_id
        transitarea area [ {
            retransmitinterval vl_time ;
            transitdelay vl_time ;
            priority vl_priority ;
            hellointerval vl_time ;
            routerdeadinterval vl_time ;
            pollinterval vl_time ;
            passive ;
            advertise-subnet on | off ;
            auth [none | simple auth_key | md5 md5-keyset] ;
        } ] ;
    } ;
```

```
} ] ;
```

More detailed descriptions of these commands can be found on page 95 of the *Command Reference Guide*. See “Exporting to OSPF ASE and NSSA” on page 151 for information about exporting and OSPF.

12.3 OSPF Sample Configurations

12.3.1 Example 1, Host Configuration

The simplest configuration for a host user is the following, which will set GateD into the backbone area specified for all interfaces. The GateD OSPF has defaults for a host that will not allow it to become a designated router (DR) for OSPF.

```
ospf yes;
```

The simplest configuration for a host user in an area outside the backbone is:

```
ospf yes {
    area 0.0.0.2; {
        interface all;
    };
};
```

12.3.2 Example 2, Router Configurations

The same simplest configurations can also be used for UNIX system running as a router. The following configuration is for a router in the backbone.

```
ospf yes {
    priority 1;
    backbone {
        interface all;
    };
};
```

The following gives the same configuration as above.

```
ospf yes { priority 1; };
```

The following configuration is for a router in area 0.0.0.2.

```
ospf yes {
    priority 1;
    area 0.0.0.2 {
        interface all;
    };
};
```

The following configuration is for a simple border router.

```
ospf yes {
    priority 1;
    backbone {
        interface fxp0;
    };
    area 0.0.0.1 {
        interface fxp1;
    };
};
```

Use area ranges to reduce the amount of routing information in the OSPF domain. In this example, area 0.0.0.1 is the only area with 192.168.x/24 networks in it. By specifying a network range, only a single LSA is announced to the backbone (and thus to other areas) advertising the larger 192.168/16 route. The following configuration is for a border router with summarizing area range.

```
ospf yes {
    priority 1;
    backbone {
        interface fxp0;
    };
    area 0.0.0.1 {
        networks {
            192.168 masklen 16;
        };
        interface fxp1;
    };
};
```

To reduce the amount of routing information in a single area, make it a stub area. Stub areas do not receive LSAs for external routes being readvertised in OSPF. Normally, you would also originate a default summary route into the area, so that internal routers have a route to networks outside of the area. The following configuration is for a border router attaching to a stub area and injecting a default route.

```
ospf yes {
    priority 1;
    backbone {
        interface fxp0;
    };
```

```
        area 0.0.0.1 {
            stub cost 1;
            interface fxp1;
        };
    };
```

To further reduce the amount of routing information, when using stub areas, you can filter all (or some subset) of the summary (except the generated default). Be sure to specify the **cost 1** part of the stub statement so that a default route is generated for the routers in the stub area. The following configuration is for a border router attaching to stub area, injecting a default route and filtering all summary.

```
    ospf yes {
        priority 1;
        backbone {
            interface fxp0;
        };
        area 0.0.0.1 {
            stub cost 1;
            summary-filters { all; };
            interface fxp1;
        };
    };
```

12.4 Authentication

By definition, all OSPF protocol exchanges are authenticated; however, one method of authentication is none. Authentication can help to guarantee that routing information is imported only from trusted routers. A variety of authentication schemes can be used, but a single scheme must be configured for each network. The use of different schemes enables some interfaces to use much stricter authentication than others. The three authentication schemes available are: none, simple, and MD5.

No Authentication

When no authentication is required, use authentication type **none**. To use authentication type **none**, add the following line to the appropriate OSPF interface statements.

```
    auth none ;
```

Simple Authentication Key

When you want to keep certain routers from exchanging OSPF packets, use the simple form of authentication. The interfaces that the packets are to be sent on still need to be trusted, because the key will be placed in the packets and can be seen by anyone with access to the network. To specify authentication type **simple**, add the following line to your OSPF interface statements:

```
auth simple "auth-key"
```

where *auth-key* is a quoted string of up to eight characters.

MD5 Authentication

When you do not trust other users of your network, use MD5 authentication. The system works by using shared secret keys. Because the keys are used to sign the packets with an MD5 checksum, they cannot be forged or tampered with. Because the keys are not included in the packet, snooping the key is not possible. Users of the network can still snoop the contents of packets, however, because the packets are not encrypted.

GateD's MD5 authentication is compliant with the specification in OSPF RFC 2328. This specification uses the MD5 algorithm and an authentication key of up to 16 characters. RFC 2328 allows multiple MD5 keys per interface. Each key has two associated time ranges.

To specify a single MD5 key on an interface, add the following to the appropriate OSPF interface statements:

```
auth md5 md5-keyset
```

where *md5-keyset* is:

```
key auth-key id id-number [ {
    [start-generate date-time;]
    [stop-generate date-time;]
    [start-accept date-time;]
    [stop-accept date-time;]
}];
```

where *auth-key* is a 1- to 16-character string in double quotes, *id-number* is an integer between 1 and 255, and *date-time* is in the format YYYY/MM/DD HH:MM. (If any time fields are used, all are required.)

If no value is given for the time ranges, the default values are:

- key is always generated
- key is always accepted

Thus, if you always want your key to be accepted, simply specify a sequence such as:

```
auth md5 key "mykeyone" id 1;
```

To specify multiple MD5 keys on an interface, add the following to the appropriate OSPF interface statements:

```
auth md5 {
    md5-key
    md5-key
        .
        .
        .
    md5-key
} ;
```

where *md5-key* is as specified above.

For example, two routers may start out generating key 1 and want to switch to key 2 at 6:00 GMT. In order to make the transition between keys easier, the routers agree to stop generating key 1 at 6:00 GMT but accept key 1 until 6:10 GMT. Key 2 is accepted ten minutes before the planned switch time (i.e., 5:50 GMT). These overlapping ranges allow the clocks on the routers to be slightly out of sync. This sequence of keys would be specified by:

```
auth md5 {
    key "mykeyone" id 1 {
        stop-generate 2001/05/01 06:00;
        stop-accept 2001/05/01 06:10;
    };
    key "mykeytwo" id 2 {
        start-generate 2001/05/01 06:00;
        start-accept 2001/05/01 05:50;
    };
};
```