This document is an installation guide for machines comprising the LANDNET proof of concept.

After configuring each component of the system, details are given on how to test to ensure that particular component is functioning correctly.

Contents

Bubblephone Limited
Sussex Innovation Centre
Science Park Square
University of Sussex
Brighton BN1 9SB
Tel +44 (0) 1273 704 535
http://www.bubblephone.com/

Topology

Two equivalent networks are set up; using Cisco IOS for confirmation by industry standard, and Quagga for the proof-of-concept system.

In both cases, all three nodes form a single OSPF area, the requisite Area 0.

All nodes are fully-connected in the ideal:

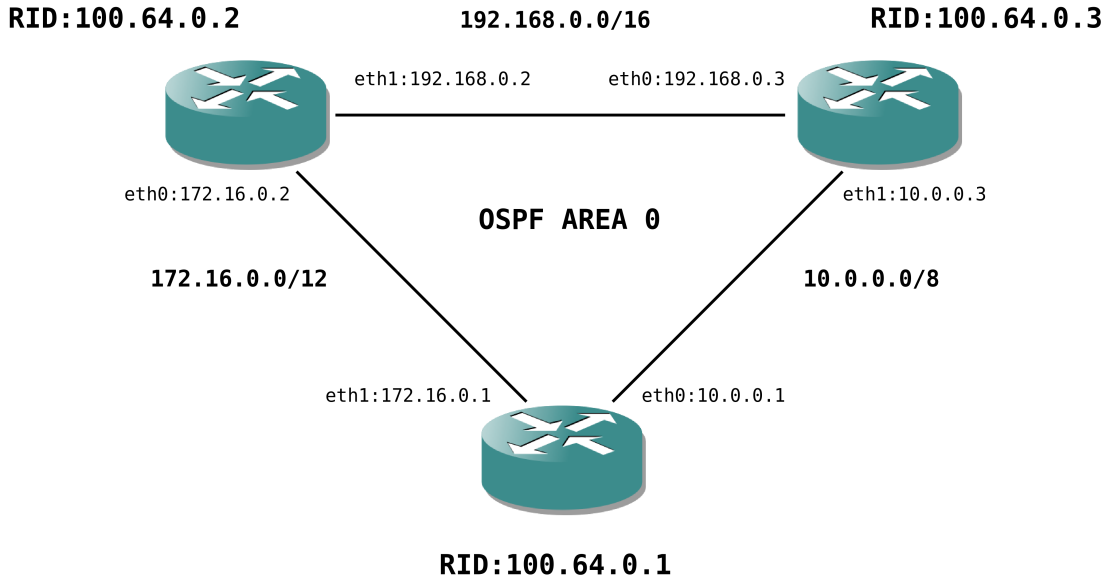

Figure 1. Quagga Numbering

Individual links may be disabled per connectivity requirements, to simulate loss of media.

The numbering scheme described here is designed for clarity of various network topologies during development. The function of the system does not depend on any particular numbering scheme; for production networks, other numbering schemes may be used.

For this scheme, each node in the network is given a unique numeric host name. This number is taken as a unique ID, forming the basis of the IP and Router-ID (RID) allocation. Although RIDs are traditionally written in `0.0.0.0` form, they are not IP addresses. In our allocation scheme, we use these arbitrary numbers to express “overlay” IP addresses in the `100.64/10` Carrier-Grade NAT range.

SNMP

1. Install Net-SNMP:

```
# apt-get install libsnmp-mib-compiler-perl libsnmp-perl libsnmp-session-perl \
                  snmp snmpd snmptrapfmt snmptt
```

2. Install /etc/snmp/snmpd.conf (see appendix) [1]
3. /etc/init.d/snmpd restart

Optionally: For human readable OID names, which are helpful for debugging:

4. Add `non-free` to the package repositories (`/etc/apt/sources.list` for Debian)
5. Install the MIB downloader and run it:

```
# apt-get install snmp-mibs-downloader
# download-mibs
```

6. Comment out the `mibs :` line in `/etc/snmp/snmp.conf` [2]
7. Restart **`snmpd`**:

```
# /etc/init.d/snmpd restart
```

Confirm **`snmpd`** is indeed running:

```
# ps ww -C snmpd
  PID TTY      STAT   TIME COMMAND
11499 ?        S      0:03 /usr/sbin/snmpd -Lsd -Lf /dev/null -u snmp -g snmp -p
/var/run/snmpd.pid
```

If it's not, then that's likely due to failing to parse `snmpd.conf`; see syslog for details.

You should now be able to bulkwalk **`snmpd`**. It's handy to alias this, for convenience due to the command length:

```
% alias bw='snmpbulkwalk -v3 -l authNoPriv -a SHA -x DES -u broen -A password -OSa'
% bw localhost
```

This should give the entire OID tree. **`snmpbulkwalk`** will rather unhelpfully time out rather than reporting an error (probably due to using UDP), if **`snmpd`** is not running.

To walk a sub-tree, give the OID for that branch:

```
% bw localhost UCD-SNMP-MIB::prTable
```

---

1 For earlier versions of Net-SNMP, comment out the `agentAddress` line in `snmpd.conf`.

2 Note `/etc/snmp/snmp.conf`, not `snmpd.conf`. If you don't have one, then never mind.

SMUX

Enable smux for Net-SNMP:

1. Remove the `-I -smux` argument from SNMPDOPTS in `/etc/default/snmpd`:

```
-SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -g snmp -I -smux -p /var/run/snmpd.pid'
+SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -g snmp -p /var/run/snmpd.pid'
```

   The default was `-I -smux` for Debian. Note removing it is not the same as changing it to `-I smux` as the list of modules enabled by `-I` is explicit.

2. Restart **snmpd**:

```
# /etc/init.d/snmpd restart
```

3. Confirm Net-SNMP is listening for SMUX (well-known port 199) :

```
% netstat -an | grep 199
tcp        0      0 127.0.0.1:199           0.0.0.0:*               LISTEN
```

Test SNMP traps:

1. Install `/etc/snmp/snmptrapd.conf` (see appendix)
2. Edit `/etc/default/snmpd` and enable **snmptrapd**:

```
TRAPDRUN=yes
```

3. Restart **snmpd**:

```
# /etc/init.d/snmpd restart
```

snmptrapd should now be logging traps to **syslog**. **snmpd** itself generates a few traps,

for example a SNMPv2-MIB::coldStart trap on startup:

```
# /etc/init.d/snmpd restart
# grep snmptrapd /var/log/syslog
Aug 24 18:10:00 q1 snmptrapd[3253]: 2013-08-24 18:10:00 0.0.0.0(via UDP:
[127.0.0.1]:44942->[127.0.0.1]) TRAP, SNMP v1, community
public#012#011iso.3.6.1.4.1.8072.3.2.10 Cold Start Trap (0) Uptime: 0:00:00.05#012
```

Quagga

SNMP isn't enabled by default for Debian's Quagga package. So we need to build Quagga with SNMP support.

1. Get the source: [3] [4]

```
# cd /usr/local/src
# apt-get source quagga
# apt-get build-dep quagga
```

2. Install build dependencies. Note the exact dependencies differ from system to system:

```
# apt-get install libsnmp-dev libsnmp9-dev libsnmp9 libsnmp-base \
                  debhelper dpkg-dev snmpd snmp
```

3. Edit `quagga-*/debian/quagga.preinst` and remove the `grep ^smux` block near the end of the file.
4. Build:

```
# cd /usr/local/src
# export WANT_SNMP=1
# apt-get -b source quagga
```

5. This should produce a `.deb` package. Install it:

```
# dpkg -i quagga_*.deb
```

   The Quagga MIBs should also be included in the package above, installed as `/usr/share/snmp/mibs/GNOME-PRODUCT-ZEBRA-MIB`

3 Some Debian-derived distributions do not include source repositories by default, in which case you will need to add one by appending a `deb-src` URL to `/etc/apt/sources.list`. Don't forget to `apt-get update` after adding a repository. For example, for Mint 9, append: `deb-src http://archive.ubuntu.com/ubuntu lucid main restricted universe multiverse`

4 Mint 9 in particular appears to lack dpkg-dev. You can `apt-get install` it, or get it from: `http://community.linuxmint.com/software/view/dpkg-dev`

Configure Quagga:

1. Make somewhere for the logs to go:

```
# mkdir -p /var/log/quagga
```

2. Edit `/etc/quagga/debian.conf` and add `-f` options:

```
zebra_options="--daemon -A 127.0.0.1 -f /etc/quagga/zebra.conf"
ospfd_options="--daemon -A 127.0.0.1 -f /etc/quagga/ospfd.conf"
```

3. Edit `/etc/quagga/daemons` to enable **`zebra`**, **`ospfd`**:

```
zebra=yes
bgpd=no
ospfd=yes
ospf6d=no
ripd=no
ripngd=no
isisd=no
babeld=no
```

4. Install `/etc/quagga/zebra.conf` (see appendix)
5. Install `/etc/quagga/ospfd.conf` (see appendix)
6. Restart Quagga:

```
# /etc/init.d/quagga restart
```

You should now be able to query for SNMP OSPF status:

```
# snmpbulkwalk -v3 -l authNoPriv -a SHA -x DES -u broen -A password -Osa localhost \
    OSPF-MIB::ospfRouterId.0
```

Which should give SNMP data delivered from Quagga over SMUX:

```
OSPF-MIB::ospfRouterId.0 = IpAddress: 100.64.0.12
```

Here `OSPF-MIB::ospfRouterId.0` is shorthand for `.iso.org.dod.internet.mgmt.mib-2.ospf.ospfGeneralGroup.ospfRouterId.0` which in numeric form is: `.1.3.6.1.2.1.14.1.1.0`

VTYSH

1. Grant permissions for **vtysh**:

```
# chown quagga:quagga /etc/quagga/*.conf
# chmod 640 /etc/quagga/*.conf
```

2. Debian overrides the default pager for **vtysh** to **less**, which gives an annoying prompt after every command. To set this back to the **vytsh** default, which only prompts when necessary:

```
# echo VTYSH_PAGER=more >> /etc/environment
# export VTYSH_PAGER=more
```

You should now be able to use **vtysh** to configure and write changes:

```
# vtysh

Hello, this is Quagga (version 0.99.21).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

q1# conf term
q1(config)#
q1(config)# ^Z
q1# wr ite
Building Configuration...
Configuration saved to /etc/quagga/zebra.conf
Configuration saved to /etc/quagga/ospfd.conf
[OK]
q1#
```

OSPF

After the above configuration, **ospfd** will begin to identify its currently reachable neighbours by OSPF. You should be able to use **vtysh** to query Quagga for its current OSPF state:

```
q1# show ip ospf
 OSPF Routing Process, Router ID: 100.64.0.10
 Supports only single TOS (TOS0) routes
 This implementation conforms to RFC2328
 RFC1583Compatibility flag is disabled
 OpaqueCapability flag is disabled
 Initial SPF scheduling delay 200 millisec(s)
 Minimum hold time between consecutive SPFs 1000 millisec(s)
 Maximum hold time between consecutive SPFs 10000 millisec(s)
 Hold time multiplier is currently 1
 SPF algorithm last executed 7m35s ago
 SPF timer is inactive
 Refresh timer 10 secs
 Number of external LSA 0. Checksum Sum 0x00000000
 Number of opaque AS LSA 0. Checksum Sum 0x00000000
 Number of areas attached to this router: 1
 All adjacency changes are logged

 Area ID: 0.0.0.0 (Backbone)
   Number of interfaces in this area: Total: 2, Active: 2
   Number of fully adjacent neighbors in this area: 1
   Area has no authentication
   SPF algorithm executed 3 times
   Number of LSA 3
   Number of router LSA 2. Checksum Sum 0x00003cc7
   Number of network LSA 1. Checksum Sum 0x00005fca
   Number of summary LSA 0. Checksum Sum 0x00000000
   Number of ASBR summary LSA 0. Checksum Sum 0x00000000
   Number of NSSA LSA 0. Checksum Sum 0x00000000
   Number of opaque link LSA 0. Checksum Sum 0x00000000
   Number of opaque area LSA 0. Checksum Sum 0x00000000

q1#
```

Here we can see the state for the single OSPF area, Area 0 (written by Quagga in its equivalent notation, 0.0.0.0). The list of current neighbours may be shown by **vtysh**:

```
q1# show ip ospf neighbor

Neighbor ID  Pri  State       Dead Time Address    Interface       RXmtL RqstL DbsmL
100.64.0.3     1  Full/DR     36.256s   10.0.0.3   eth0:10.0.0.1       0     0     0
100.64.0.2     1  Full/Backup 37.704s   172.16.0.2 eth1:172.16.0.1     0     0     0
q1#
```

Here you can see q1 has two neighbours; each lists its RID, the IP Address by which it may be reached, and the local interface to which it is attached.

This information changes as **ospfd** updates per the currently available neighbours, depending on which machines are physically accessible at any given time.

As neighbour adjacency changes, **ospfd** will create and destroy IP routes per their best metrics. The OSPF database gives a convenient summary for the current state:

```
q1# show ip ospf database

       OSPF Router with ID (100.64.0.1)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
100.64.0.1      100.64.0.1        96 0x80000012 0xafe8 2
100.64.0.2      100.64.0.2        34 0x80000023 0x4450 2
100.64.0.3      100.64.0.3      1475 0x80000008 0x7237 2

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
172.16.0.1      100.64.0.1       126 0x80000005 0x75e0
10.0.0.3        100.64.0.3      1305 0x80000003 0x5cf5


q1#
```

Confirm Quagga is creating OSPF routes:

```
q1# show ip ospf route
============ OSPF network routing table ============
N    172.16.0.0/12           [10] area: 0.0.0.0
                             directly attached to eth1
N    10.0.0.0/8              [10] area: 0.0.0.0
                             directly attached to eth0
N    192.168.0.0/16          [20] area: 0.0.0.0
                             via 10.0.0.3, eth0
N    192.168.0.0/16          [20] area: 0.0.0.0
                             via 172.16.0.2, eth1

============ OSPF router routing table =============

============ OSPF external routing table ===========

q1#
```

If all went well, you should be able to ping IPs from interfaces which are not directly connected. For example, in **sh** on q1:

```
q1# ping -c3 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_req=1 ttl=64 time=0.483 ms
64 bytes from 192.168.0.2: icmp_req=2 ttl=64 time=0.499 ms
64 bytes from 192.168.0.2: icmp_req=3 ttl=64 time=0.609 ms

--- 10.2.3.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.483/0.530/0.609/0.059 ms
q1#
```

Routing

1. Enable IP forwarding:

```
# sysctl -w net.ipv4.ip_forward=1
```

2. Append to `/etc/sysctl.conf` for the next boot:

```
net.ipv4.ip_forward = 1
```

Iptables

1. Install the shell trap handler:

```
# wget http://www.bubblephone.com/pub/ospf-handler.sh
# mv ospf-handler.sh /etc/snmp
# chmod 755 /etc/snmp/ospf-handler.sh
```

2. Set `$RID_SELF` in `/etc/snmp/ospf-handler.sh` to this router's own RID.
3. Create a rule to map to localhost, where `100.64.0.x` is this router's own RID. This avoids needing to wait for an initial OSPF neighbour to be found:

```
# iptables -t nat -I OUTPUT --dest 100.64.0.x -j DNAT --to-dest 127.0.0.1
```

4. Create rules for source NATing outgoing traffic for each of the OSPF interfaces:

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

5. Add the same **`iptables`** commands to `/etc/rc.local` for the next boot.

OSPF traps should now be handled. You can confirm this by doing something which causes OSPF to update a link, such as bringing an interface up. Note that just bringing an interface up will not necessary cause a trap if that router is already reachable by more desirable means, as determined by OSPF's metrics for the link. So you can take down all interfaces for a given router, just to be sure:

```
q2# ifconfig eth1 down
q2# ifconfig eth0 down
q2# ifconfig eth0 up
```

`ospf-handler.sh` logs to syslog. At the same time as bringing interfaces up and down, watch for log messages:

```
q1# tail -f /var/log/syslog | grep TRAP:
Sep 20 23:39:01 q1 TRAP: OSPF RouterID=100.64.0.2 IPAddr=172.16.0.2 IfState=8
^C
```

The end result should be that when OSPF links come up, respective DNAT rules are created by **iptables**. Those rules rewrite packets to the current 10/8 IP for that particular router. **ospf-handler.sh** logs about creating and removing these rules, too:

```
q1# tail -f /var/log/syslog | grep TRAP:
Sep 20 23:39:01 q1 TRAP: DNAT Remove 100.64.0.2 -> 172.16.0.2 #1
Sep 20 23:39:01 q1 TRAP: DNAT Create 100.64.0.2 -> 172.16.0.2
^C
```

Confirm that **iptables** does indeed have the rules the trap handler creates:

```
q1# iptables -t nat -L | grep 100.64
DNAT       all  --  anywhere             100.64.0.2           to:172.16.0.2
DNAT       all  --  anywhere             100.64.0.3           to:10.0.0.3
q1#
```

Polling

Although the OSPF link state database is distributed to all nodes, Quagga does not raise SNMP traps for non-adjacent OSPF state changes. This section provides a workaround for that, by polling for those changes, then raising traps in much the same way as if Quagga did so itself.

1. Install the OSPF polling script:

```
# wget http://www.bubblephone.com/pub/ospf-poll.sh
# mv ospf-poll.sh /etc/snmp
# chmod 755 /etc/snmp/ospf-poll.sh
```

2. Run it:

```
# nohup /etc/snmp/ospf-poll.sh &
```

3. Add to the middle of **`/etc/rc.local`** for the next boot:

```
nohup /etc/snmp/ospf-poll.sh > /dev/null &
```

4. Disable handling for Quagga's traps, and enable handling for traps from **`ospf-poll.sh`** instead, by editing `/etc/snmp/snmptrapd.conf`:

```
# XXX: disabled for ospf-poll.sh workaround
# traphandle .1.3.6.1.4.1.3317.1.2.5 /etc/snmp/ospf-handler.sh

traphandle .1.3.6.1.2.1.14.16.2.16 /etc/snmp/ospf-handler.sh
```

Confirm the polling script is observing OSPF link state changes correctly, by bringing up and down a few interfaces on various nodes, and watching for its messages in syslog at the same time:

```
# tail -f /var/log/syslog | grep POLL:
Sep 21 20:23:41 q1 POLL: 100.64.0.2 move  172.16.0.2 -> 192.168.0.2
Sep 21 21:14:04 q1 POLL: 100.64.0.2 down
Sep 21 21:14:11 q1 POLL: 100.64.0.1 down
Sep 21 21:14:13 q1 POLL: 100.64.0.3 move  10.0.0.3 -> 192.168.0.3
Sep 21 21:14:13 q1 POLL: 100.64.0.1 up    10.0.0.1
Sep 21 21:14:14 q1 POLL: 100.64.0.2 up    192.168.0.2
Sep 21 21:14:23 q1 POLL: 100.64.0.3 move  192.168.0.3 -> 10.0.0.3
```

Overlay Addressing

Whenever a particular OSPF router is up (reachable by whatever `10/8` IP OSPF decides is best), you should be able to ping it by its Router ID serving as an “overlay” address:

```
q1# ping -c3 100.64.0.2
PING 100.64.0.2 (100.64.0.2) 56(84) bytes of data.
64 bytes from 100.64.0.2: icmp_req=1 ttl=64 time=1.38 ms
64 bytes from 100.64.0.2: icmp_req=2 ttl=64 time=1.10 ms
64 bytes from 100.64.0.2: icmp_req=3 ttl=64 time=1.30 ms

--- 100.64.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.100/1.261/1.382/0.118 ms
q1#
```

And that should remain true as the OSPF topology changes, as long as the router is reachable somehow. Note the change in `icmp_req` and `ttl` fields:

```
64 bytes from 100.64.0.2: icmp_req=137 ttl=64 time=1.25 ms
64 bytes from 100.64.0.2: icmp_req=138 ttl=64 time=1.07 ms
64 bytes from 100.64.0.2: icmp_req=141 ttl=63 time=0.969 ms
64 bytes from 100.64.0.2: icmp_req=142 ttl=63 time=0.784 ms
```

Here the first two pings there are routed to a directly-adjacent neighbour (as can be seen by the TTL of 64).

Then an interface was taken down, and a couple of packets get dropped between ICMP sequence numbers 138 and 141, while OSPF reconverges. Then when the OSPF database has distributed a new configuration, **`ospf-poll.sh`** observes the change, produces relevant SNMP traps, and **`ospf-handle.sh`** sets up a new redirection rule for `100.64.0.2` accordingly.

Then the second two pings go via that new route, and so have a different TTL, because (in this case) they have travelled an extra hop.

Caveats

snmpd logs an entry for every SNMP request received from a client. For example:

```
# grep snmp /var/log/syslog
Aug 23 18:37:08 debian snmpd[5401]: Connection from UDP: [127.0.0.1]:52954->[127.0.0.1]
Aug 23 18:37:08 debian snmpd[5401]: Connection from UDP: [127.0.0.1]:52954->[127.0.0.1]
Aug 23 18:37:08 debian snmpd[5401]: Connection from UDP: [127.0.0.1]:52954->[127.0.0.1]
…
```

Although useful for debugging, that excessive logging would cause huge log files for production use. Since Net-SNMP provides no way to disable that, we have two options to suppress these messages:

- Have Syslog filter them; various Syslog implementations are capable of matching patterns and conditionally disregarding them.
- Patch Net-SNMP by commenting out the offending call to `snmp_log()` in `netsnmp_agent_check_packet()` in `agent/snmp_agent.c`

The workaround for polling Quagga is obviously undesirable. A more natural implementation would be to have Quagga push out that information asynchronously, as SNMP traps. This could be added to Quagga relatively easily.

Appendix A: Net-SNMP configurations

`/etc/snmp/snmpd.conf`:

```
# Listen on all interfaces, IPv4 only
agentAddress udp:161

# read-only access for SNMPv3 encrypted requests
createUser broen SHA "password" DES
rouser broen priv

sysLocation Somewhere
sysContact  Tom F <tom.flavel@bubblephone.com>

# application | end-to-end
sysServices 72

# UCD-SNMP-MIB::prTable
# 1 min, 1 max of the following daemons
proc zebra 1 1
proc ospfd 1 1

# UCD-SNMP-MIB::laTable
load 12 10 5

# Traps, v1, v2c and v2c INFORMs respectively
trapsink   localhost public
trap2sink  localhost public
informsink localhost public

iquerySecName broen
rouser        broen

# trap on linkUp/Down
linkUpDownNotifications yes

# SMUX
smuxsocket 127.0.0.1

# SNMPv2-SMI::enterprises.3317.1.2
# depending on your version of net-snmp, these OIDs may need a trailing '.'
# e.g.: smuxpeer .1.3.6.1.4.1.3317.1.2.1. quagga_zebra_passw
smuxpeer .1.3.6.1.4.1.3317.1.2.1 quagga_zebra_passw
smuxpeer .1.3.6.1.4.1.3317.1.2.2 quagga_bgpd_passw
smuxpeer .1.3.6.1.4.1.3317.1.2.5 quagga_ospfd_passw

# include everything under .1 (0x80 is the first bit significant)
view all included .1 0x80
```

/etc/snmp/snmptrapd.conf:

```
outputOption n

# not for production use; this is just to confirm traps are working.
disableAuthorization yes

# XXX: disabled for ospf-poll.sh workaround
# traphandle .1.3.6.1.4.1.3317.1.2.5 /etc/snmp/ospf-handler.sh

traphandle .1.3.6.1.2.1.14.16.2.16 /etc/snmp/ospf-handler.sh
```

Appendix B: Quagga configurations

q1:/etc/quagga/zebra.conf

```
hostname q1
password password
enable password zebra
log file /var/log/quagga/zebra.log
service advanced-vty
!
interface eth0
 ip address 10.0.0.1/8
 ipv6 nd suppress-ra
 link-detect
 no shutdown
!
interface eth1
 ip address 172.16.0.1/12
 ipv6 nd suppress-ra
 link-detect
 no shutdown
!
interface lo
!
smux peer 1.3.6.1.4.1.3317.1.2.1 quagga_zebra_passw
line vty
```

q2:/etc/quagga/zebra.conf

```
hostname q2
password password
enable password zebra
log file /var/log/quagga/zebra.log
service advanced-vty
!
interface eth0
 ip address 172.16.0.2/12
 ipv6 nd suppress-ra
 link-detect
 no shutdown
!
interface eth1
 ip address 192.168.0.2/16
 ipv6 nd suppress-ra
 link-detect
 no shutdown
!
interface lo
!
smux peer 1.3.6.1.4.1.3317.1.2.1 quagga_zebra_passw
line vty
```

q1:/etc/quagga/ospfd.conf

```
log file /var/log/quagga/ospfd.log
!
interface eth0
 ip ospf network broadcast
 ip ospf dead-interval minimal hello-multiplier 10
!
interface eth1
 ip ospf network broadcast
 ip ospf dead-interval minimal hello-multiplier 10
!
interface lo
!
router ospf
 ospf router-id 100.64.0.1
 log-adjacency-changes detail
 network 10.0.0.0/8 area 0.0.0.0
 network 192.168.0.0/16 area 0.0.0.0
 network 172.16.0.0/12 area 0.0.0.0
 timers throttle spf 100 100 200
 passive-interface default
 no passive-interface eth0
 no passive-interface eth1
!
smux peer 1.3.6.1.4.1.3317.1.2.5 quagga_ospfd_passw
line vty
```

q2:/etc/quagga/ospfd.conf

```
log file /var/log/quagga/ospfd.log
!
interface eth0
 ip ospf network broadcast
 ip ospf dead-interval minimal hello-multiplier 10
!
interface eth1
 ip ospf network broadcast
 ip ospf dead-interval minimal hello-multiplier 10
!
interface lo
!
router ospf
 ospf router-id 100.64.0.2
 log-adjacency-changes detail
 network 10.0.0.0/8 area 0.0.0.0
 network 192.168.0.0/16 area 0.0.0.0
 network 172.16.0.0/12 area 0.0.0.0
 timers throttle spf 100 100 200
 passive-interface default
 no passive-interface eth0
 no passive-interface eth1
!
smux peer 1.3.6.1.4.1.3317.1.2.5 quagga_ospfd_passw
line vty
```

q3:/etc/quagga/zebra.conf

```
hostname q3
password password
enable password zebra
log file /var/log/quagga/zebra.log
service advanced-vty
!
interface eth0
 ip address 192.168.0.3/16
 ipv6 nd suppress-ra
 link-detect
 no shutdown
!
interface eth1
 ip address 10.0.0.3/8
 ipv6 nd suppress-ra
 link-detect
 no shutdown
!
interface lo
!
smux peer 1.3.6.1.4.1.3317.1.2.1 quagga_zebra_passw
line vty
```

q3:/etc/quagga/ospfd.conf

```
log file /var/log/quagga/ospfd.log
!
interface eth0
 ip ospf network broadcast
 ip ospf dead-interval minimal hello-multiplier 10
!
interface eth1
 ip ospf network broadcast
 ip ospf dead-interval minimal hello-multiplier 10
!
interface lo
!
router ospf
 ospf router-id 100.64.0.3
 log-adjacency-changes detail
 network 10.0.0.0/8 area 0.0.0.0
 network 192.168.0.0/16 area 0.0.0.0
 network 172.16.0.0/12 area 0.0.0.0
 timers throttle spf 100 100 200
 passive-interface default
 no passive-interface eth0
 no passive-interface eth1
!
smux peer 1.3.6.1.4.1.3317.1.2.5 quagga_ospfd_passw
line vty
```