```
% Anycast lab

**Anycast service configuration**

We are going to create a class-wide anycast resolver infrastructure.

The goal is to configure your resolver to respond to an anycast
IP address (here: 20.20.20.20/32), and use an IGP (OSPF) to
announce availability of this IP on the local network.

# Host configuration - on your `resolv` server

The host configuration is detailed below.

## Creating looppback

Start by creating a new interface on your host, called `lo1`.

    $ sudo ifconfig lo1 create

## Anycast address

we're going to assign the anycast address to be used in the classroom
(20.20.20.20) to this interface:

    $ sudo ifconfig lo1 20.20.20.20/32

Note: to make this permanent, we'd add the following lines to
`/etc/rc.conf`:

    cloned_interfaces="lo1"
    ifconfig_lo1="20.20.20.20/32"

Test that you can ping this IP locally:

    $ ping 20.20.20.20

## Enabling the routing daemon

We now need to enable OSPF on this host, so that it will announce
the prefix (20.20.20.20/32) into the network.

We'll use Quagga (http://www.nongnu.org/quagga/) for this.

To enable the routing control daemon (zebra) and the OSPF daemon (`ospfd`),
add the following lines to `/etc/rc.conf`:

    quagga_daemons="ospfd zebra"
    quagga_enable="YES"

Configure quagga:

    $ cd /usr/local/etc/quagga
    $ sudo touch ospfd.conf zebra.conf
    $ sudo chown quagga *
    $ ls -l

Start the routing services:
    $ sudo service quagga start
    $ ps ax | grep ospfd
    $ ps ax | grep zebra
```

## Configuring OSPF to announce the IP

The quagga `vtysh` connects you to the Quagga command line interface,
which is very similar to Cisco IOS.

```
    $ sudo vtysh
```

You should now be presented with a prompt similar to this:

```
    resolv.grpXX.dns.nsrc.org#
```

Enter configuration mode:

```
    # conf terminal
```

You should now see:

```
    resolv.grpXX.dns.nsrc.org(config)#
```

Let's configure OSPF:

```
    # router ospf
    # ospf router-id 10.20.x.3
    # network 10.20.0.0/16 area 0
    # network 20.20.20.20/32 area 0
    # exit
    # exit
    # write mem
```

# Reconfigure unbound

Unbound needs to be told to use the right address for the reply.

Edit the file `/usr/local/etc/unbound/unbound.conf`, and in the
server category, add the following statement:

```
    interface-automatic: yes
```

Save the file and exit, then restart unbound:

```
    # service unbound restart
```

Note: this is required to make sure that the source IP address Unbound will
use to replying to the client, is the same that it received the request on.

# Router configuration (this is done by the instructor)

The instructor will configure a router to listen to the OSPF announcements,
and on the projector we'll look as the different groups configure the anycast
address, and the OSPF routing table is updated.

As the class progresses, the instructor will update the OSPF view on the
projector:

~~~
~~~

```
*>  20.20.20.20/32, Intra, cost 11, area 0
      via 10.10.15.1, FastEthernet0/0
      via 10.10.20.3, FastEthernet0/0
~~~
```

## Testing anycast!

On your OSPF process, run the following commands (use `sudo vtysh`)

```
    # show ip route ospf
```

You will only see your own route (directly connected), being advertised
to the world:

```
~~~
O   10.10.0.0/16 [110/10] is directly connected, eth0, 00:04:35
O   20.20.20.20/32 [110/10] is directly connected, lo1, 00:04:35
~~~
```

Now, it's time to use this anycast platform we have built.

From one of your servers (auth or resolv), run the following command:

```
    dig @20.20.20.20 www.YOURTLD
```

If everything goes well, you will be getting the answer - however,
we don't know which server answered.

To find out, let's analyze the traffic as it arrives on your servers.

We will do this in class, using tcpdump and ping.

In a new window, on your resolver, run tcpdump:

```
    $ sudo tcpdump -n -i eth0 port 53 or icmp
```

Let the instructor know when you have done this.

We will then proceed to:

- ping 20.20.20.20 - who in the class is receiving the traffic ?
- run `dig @20.20.20.20 www.YOURTLD` - same question as above

Observe the sequence numbers in the tcpdump output.

You can run these tests yourself:

- ping and dig to 20.20.20.20 from another server in the class - one
  NOT configured to run the anycast service.

- Use traceroute (or mtr -n) to 20.20.20.20 and see the path followed.

## Identifying the DNS server

As shown in class, use this command to identify which server
you are talking to:

```
    $ dig @20.20.20.20 hostname.bind txt chaos
```

# Optional

## Service monitoring

The problem with the above setup is that if the Unbound server dies
or otherwise stops responding, DNS queries to that host will go
unanswered. One way to ensure that this doesn't happen, is to remove
the OSPF prefix announcement, should the service fail.

A simple way to do this would be to run a check that periodically
checks the DNS service for availability, and would terminate the
Quagga processes (thus stopping the OSPF announcement).

In shellscript, we'd do something like:

~~~
#!/bin/sh

while true
do
    dig +short @20.20.20.20 some.hostname.to.lookup | wc -l
    if [ $? -eq 0 ]; then
        # If dig returns 0 lines, we have not found a record -
        # something is wrong!
        killall zebra
        kilall ospfd
    fi
done
~~~

This is overly simple, and there are a number of things we could
improve:

- this doesn't deal with timeouts / slow servers. We could time the
  query, and if it takes longer than, say, 500ms, and/or dig exits
  with a timeout code (error code 9), then we could decide to terminate
  OSPF

- we could consider restarting the nameserver as well, and then
  restarting OSPF

## Anycast authoritative service

What about BIND ? Can you, in collaboration with your secondaries
in the class, create anycast services for your TLDs ? You'll have
to pick a different IP (20.20.20.X/32) for each of your domains.

Tell the instructor which IP you picked.

## Security - OSPF authentication

Is it a good idea to let anyone announce prefixes on your infrastructure
network ? What would you do to limit who would be allowed to announce
an anycast prefix on your server network ?