**IK2215 Advanced Internetworking**

**ISP Project**

**Group 2**

**Group Members:**

Yiyang Tai

Khalid Khan

Wenqin Chen

Linus Mexelin

Johan Andersson

Yung-Chin Liang

Karim Kuvaja Rabhi

Saivenkata Krishna Gowtam Peesapati

Date:

10-08-2018

1. **Detail network topology**
   1. **Required equipment**

4 Cisco 7301 routers

2 HP2524 switches (for server and client subnet)

4 lab VMs

10 UTP straight cables

5 UTP cross cables

6 laptops for hosts and services

* 1. **Network map**
  2. **IP address allocation**

Our IPv4 network 10.2.0.0/20 will be divided into 8 subnets, with 10.2.1.0/24 to 10.2.5.0/24 and 10.2.9.0/24 used between routers, 10.2.6.0/24 for server subnet and 10.2.7.0/24 for client subnet. The primary link to AS1 will use subnet 10.0.16.4/30, while the private link to AS11 will use subnet 10.1.0.48/30. Our IPv6 network fd00:10:2::/48 is also divided into 8 subnets, with fd00:10:2::C001:0/112 to fd00:10:2::C005:0/112 and fd00:10:2::C009:0/112 between routers, fd00:10:2::C006:0/112 for server subnet and fd00:10:2::C007:0/112 for client subnet.

|  |  |  |  |
| --- | --- | --- | --- |
| Device | Interface | IPv4 Address | IPv6 Address |
| RTA | Fa1/1 | 10.0.16.6/30 |  |
|  | Gi0/0 | 10.2.9.2/24 | fd00:10:2::C009:2/112 |
|  | Gi0/1 | 10.2.3.1/24 | fd00:10:2::C003:1/112 |
|  | Gi0/2 | 10.2.4.1/24 | fd00:10:2::C004:1/112 |
|  | Lo0 | 10.2.0.4/32 | fd00:10:2::C000:4/128 |
| RTB | Fa1/0 | 10.1.0.51/30 |  |
|  | Gi0/0 | 10.2.1.1/24 | fd00:10:2::C001:1/112 |
|  | Gi0/2 | 10.2.5.1/24 | fd00:10:2::C005:1/112 |
|  | Lo0 | 10.2.0.2/32 | fd00:10:2::C000:2/128 |
| RTC | Fa1/0 | 10.2.7.1/24 | fd00:10:2::C007:1/112 |
|  | Gi0/0 | 10.2.1.2/24 | fd00:10:2::C001:2/112 |
|  | Gi0/1 | 10.2.2.1/24 | fd00:10:2::C002:1/112 |
|  | Gi0/2 | 10.2.4.2/24 | fd00:10:2::C004:2/112 |
|  | Lo0 | 10.2.0.1/32 | fd00:10:2::C000:1/128 |
| RTD | Fa1/0 | 10.2.6.1/24 | fd00:10:2::C006:1/112 |
|  | Gi0/0 | 10.2.3.2/24 | fd00:10:2::C003:2/112 |
|  | Gi0/1 | 10.2.2.2/24 | fd00:10:2::C002:2/112 |
|  | Gi0/2 | 10.2.5.2/24 | fd00:10:2::C005:2/112 |
|  | Lo0 | 10.2.0.3/32 | fd00:10:2::C000:3/128 |
| DNS |  | 10.2.6.2/24 | fd00:10:2::C006:2/112 |
| DHCP |  | 10.2.6.2/24 | fd00:10:2::C006:2/112 |
| Web Server |  | 10.2.6.4/24 | fd00:10:2::C006:4/112 |
| Mail Server |  | 10.2.6.3/24 | fd00:10:2::C006:3/112 |
| Client subnet |  | 10.2.7.0/24 | fd00:10:2::C007:0/112 |

**Table 1:** List of interfaces, IPv4 address and IPv6 address for routers and hosts

1. **Mandatory tasks**
   1. ***Routing functionality***
      1. **Dynamic IP routing: OSPFv3**

We are going to use OSPFv3 since it generally has more stability and functionality than distance-vector based algorithms due to computation on original data and no dependence on intermediate routers. We believe that it can provide faster and better convergence as well as network handling. Besides, OSPFv3 is compatible with both IPv4 and IPv6, and this could make the following tasks easier.

In OSPF, the packets are by default routed on the path with the least weight or in other words, the path with the least cost.  In our case, we wish to connect the clients which would be connected to the router C to reach the server connected to router D using the direct link between C and D. We would enforce the packets to take this path by decreasing the bandwidth on the links between C and B and C and A. This would cause the OSPF algorithm to pick the direct path between C and D as the default path for internal routing between the clients and the server.

* + 1. **BGP policy**

We would like to have a BGP policy in which we use our primary link to the top-tier AS and the private link to reach the neighboring AS, i.e. AS11. In case the primary link to the top-tier fails to function, we could maintain the connection to the top-tier AS through the private link and the neighboring AS.

We plan to differentiate the primary link from the private link by manipulating the LOCAL\_PREF attribute on our router. We are going to have two routers, in our case they will be RTA and RTB, running BGP. All the traffic generated from our network should be directed to other AS’s through the primary link to the top-tier AS. The private link should act as a backup, only being accessed when the primary link is not functioning. For incoming traffic from the neighboring AS’s, we would also enforce the traffic to flow through our primary link from the top-tier AS. This will be done by the multi\_exit\_disc(MED) attribute.

* 1. ***Multicast routing***

Firstly, we will run IGMP between RTC and our hosts and then PIM will be assigned among all our routers. We will assign our loopback0 IP address 10.2.0.1/32  as our OSPF router-id and RTC router’s ID 10.2.0.1/32 as a static RP on all routers since it is more stable compared to the physical interface.

* 1. ***Internet application services***
     1. **DNS**

We are planning to implement the basic DNS service with one zone. We would like to implement BIND on Linux for the DNS protocols. It is by far the most widely used DNS open source software on the Internet, providing a robust and stable platform where organizations can build distributed computing systems. Besides, BIND has evolved to be a very flexible, full-featured DNS system.

Our main DNS server, which is named ns.isp2.lab, will have IP address 10.2.6.2. In the configuration file, we will assign all routers, servers, and hosts IP addresses and names under isp2.lab domain resolvable through our DNS. For example, RTA is corresponding to lo0 10.2.0.4, and Client 1 through 20 is corresponding from 10.2.7.1 to 10.2.7.20. It will be specified in our *conf.* file of BIND. We also have IPv6 address DNS mapping as well.

All host in the same subnet(10.2) as DNS can use DNS server. The list of records that we will use are:

1. type ‘A’ record for IPv4 addresses
2. type ‘AAAA’ record for IPv6 addresses
3. type “CNAME” (canonical names) record to specify a domain name that has to be queried in order to resolve the original DNS query.
4. type “PTR” (pointer) record to map IP addresses to host names, these will be used for reverse DNS.
5. type “MX” (mail exchange) record to help other mail servers to determine where to deliver mail in our network.
   * 1. **DHCP**

We are planning that all servers should get their IP address and network mask through DHCP. We will also set up DHCP relay in the servers to provide DHCP service among server and client networks since they are not in the same subnet.

We would like to implement ISC DHCP on Linux for DHCP implementation, which is open source software that implements DHCP for connection to an IP network. It offers a complete solution for implementing DHCP servers, relay agents, and clients for small local networks to large enterprises. ISC DHCP solution supports both IPv4 and IPv6 and is suitable for use in high-volume and high-reliability applications.

We are using isc-dhcp-server to automatically assign IP addresses for client machines(The configuration is specified in *conf.* file of DHCP).

* + 1. **Web server**

The functionality of the web server is to display our project summary. To implement this we plan to run Apache on Linux, using *HTML* to produce our website on the server.

**3. Selective Tasks**

**3.1 *IPv6***

For IPv6 we want to use OSPFv3, as we expect it to provide fast and better convergence and network handling for our mail service. Also, it generally has more stability and functionality than distance-vector based algorithms due to computation on original data and no dependence on intermediate routers.

We would like to set up dual-stack IPv4 and IPv6. As above, define the policy that uses weighting metric along with path count and bandwidth to determine the path taken by the packets. The IPv6 routing policy will be the same as the IPv4, we want the traffic to go from RTC to RTD.

***3.2 Mail service (hMailServer on Windows)***

The internet application service we’ll demonstrate in the end is a mail service with a server running on Windows. We plan to install hMailServer on a Windows computer to act as our mail service server. On this host, we will implement spam protection and no open relay. The hMailServer also provides a function to assign domain name for its hosting account, so we can specify the valid accounts on our server using predefined domain name and accounts.