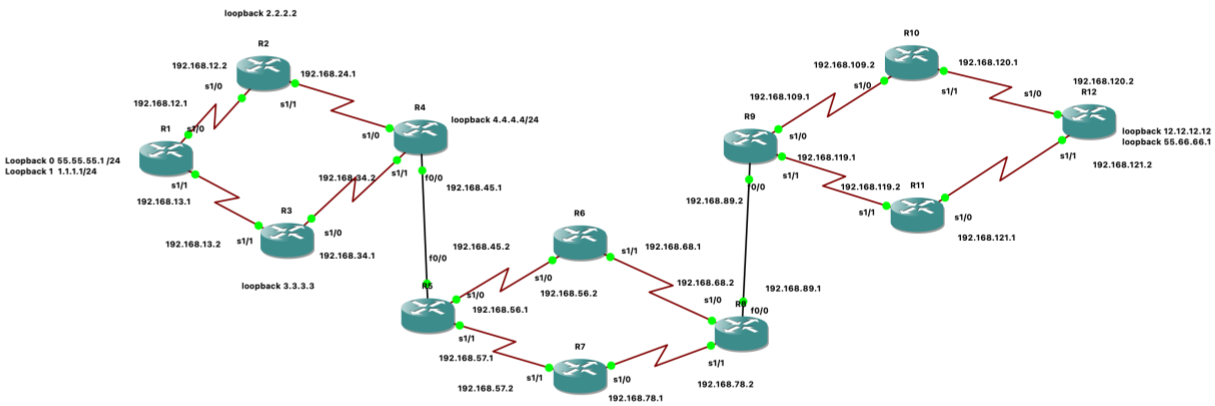
- Advanced RIP Lab –



Basic Objectives:

1. Configured and cabled the Serial and Ethernet interfaces as indicated in the diagram above.

2. Assigned IP addresses to the router interfaces using the 192.168.YY.xx/24 addressing scheme as follows:

Router 1 – 2 = 192.168.12.x Router 6 – 8 = 192.168.68.x

Router 1 – 3 = 192.168.13.x Router 7 – 8 = 192.168.78.x

Router 2 – 4 = 192.168.24.x Router 8 – 9 = 192.168.89.x

Router 3 – 4 = 192.168.34.x Router 9 – 10 = 192.168.109.x

Router 4 – 5 = 192.168.45.x Router 9 – 11 = 192.168.119.x

Router 5 – 6 = 192.168.56.x Router 10 – 12 = 192.168.120.x

Router 5 – 7 = 192.168.57.x Router 11 – 12 = 192.168.121.x

3. Configured a loopback interface on each router using the following addressing scheme: Y.Y.Y.Y/24, where Y corresponds to the router number. For example, Router 4's loopback address is 4.4.4.4/24.

RIP Objectives:

4. On Router 1 and Router 12, I created a second loopback interface on each. On Router 1, I assigned the address 55.55.55.1, and on Router 12, I assigned the address 55.66.66.1.

5. Configure RIP routing on all routers. Use ONLY RIP Version 1. Ensure that all networks are injected into the RIP process, including loopbacks. Are all networks reachable? Why or why not? What does the routing table look like?

Successfully configured RIP Protocol version 1 and ensured all networks are injected into the RIP process, including loopbacks. However, not all networks are reachable. Specifically, IP addresses 55.55.55.1 and 55.66.66.1 are unreachable while 1.1.1.1 and other networks are reachable. The main reason is RIPv1 supports only classful IP addresses and cannot distinguish between different subnets within the same class. Both 55.55.55.1/24 and 55.66.66.1/24 are seen as the same Class A network (55.0.0.0/8), creating routing conflicts and preventing proper reachability between these specific addresses.
It's like calling a phone number that two different people share.

Change the RIP version to v2. Are all networks reachable now? Why or why not? What additional command(s), if any, are necessary to make all networks reachable?

After changing the RIP version from v1 to RIP v2 on all routers, I was able to successfully reach all networks, including the previously unreachable loopback interfaces 55.55.55.1 on Router 1 and 55.66.66.1 on Router 12.

The main reason for this improvement is that RIP v2 supports classless routing (CIDR) and includes subnet mask information in its updates. This allows routers to distinguish between subnets within the same major classful network. Unlike RIP v1, RIP v2 does not automatically summarize routes at classful boundaries unless explicitly told to do so.

To ensure complete network reachability and avoid route summarization issues, I used the following additional command on each router:

```
router rip
 version 2
 no auto-summary
```

By applying no auto-summary, I prevented RIP from summarizing the loopback addresses like 55.55.55.1/24 and 55.66.66.1/24 into a single 55.0.0.0/8 route. As a result, each router now advertises the full network and subnet mask, allowing accurate route propagation across the entire topology.

At last, switching to RIP v2 along with no auto-summary successfully resolved the previous reachability issues and ensured all networks, including loopbacks, are now reachable.

Configure all interfaces to be passive. Direct RIP to only send unicast updates to its neighbors.

To enhance security and control RIP update propagation, I configured all interfaces as passive and set RIP to use unicast updates to its direct neighbors on every router.

On each router, I entered the following under RIP configuration mode:

```
router rip
 version 2
 no auto-summary
 passive-interface default
```

```
 neighbor <next-hop IP address>
```

The passive-interface default command ensures that no RIP updates are sent or received via broadcast or multicast on any interface.

The neighbor command allows RIP to send updates via unicast to specific neighbor IP addresses, restoring communication only with intended routers.

Configure md5 authentication between all neighbors. Use a key of “cisco”?

To enhance the security of RIP version 2 routing updates, I successfully configured MD5 authentication between all neighboring routers using a shared key "cisco".

Step 1: Define the authentication key chain

```
key chain rip_keys
 key 1
  key-string cisco
```

key chain rip_keys: This creates a named key chain called rip_keys that holds the secret key(s) used for authentication.

key 1: This defines the key number (you can use <0-2147483647>). All routers must use the same key number for compatibility.

key-string cisco: This is the actual shared password (secret key) used by routers to authenticate RIP updates. In this case, the key is "cisco".

Apply MD5 authentication to RIP-enabled interfaces

Example

```
interface f0/0
 ip rip authentication mode md5
 ip rip authentication key-chain rip_keys
```

ip rip authentication mode md5: This enables MD5-based authentication (instead of plain text).

ip rip authentication key-chain rip_keys: This tells the router to use the previously defined rip_keys for authenticating RIP packets on this interface.

You must apply this configuration on both sides of every RIP-enabled link to ensure updates are authenticated and accepted.

Purpose of MD5 Authentication in RIP:

Prevents unauthorized routers from injecting incorrect routing information.

Ensures that only routers with the correct key can participate in RIP route exchange.

Protects against routing table poisoning and other network attacks.

Use the rip debugging commands to ensure updates are authenticated and sent as unicasts.

```
~ — R12 — telnet 172.16.253.1 5013
R12#debug ip rip
RIP protocol debugging is on
R12#
R12#
*Jun  4 02:55:49.810: RIP: received packet with MD5 authentication
*Jun  4 02:55:49.810: RIP: received v2 update from 192.168.120.1 on Serial1/0
*Jun  4 02:55:49.814:      1.1.1.0/24 via 0.0.0.0 in 8 hops
*Jun  4 02:55:49.814:      2.2.2.0/24 via 0.0.0.0 in 7 hops
*Jun  4 02:55:49.818:      3.3.3.0/24 via 0.0.0.0 in 7 hops
*Jun  4 02:55:49.822:      4.4.4.0/24 via 0.0.0.0 in 6 hops
*Jun  4 02:55:49.822:      5.5.5.0/24 via 0.0.0.0 in 5 hops
*Jun  4 02:55:49.822:      6.6.6.0/24 via 0.0.0.0 in 4 hops
*Jun  4 02:55:49.822:      7.7.7.0/24 via 0.0.0.0 in 4 hops
*Jun  4 02:55:49.826:      8.8.8.0/24 via 0.0.0.0 in 3 hops
*Jun  4 02:55:49.826:      9.9.9.0/24 via 0.0.0.0 in 2 hops
*Jun  4 02:55:49.826:      10.10.10.0/24 via 0.0.0.0 in 1 hops
*Jun  4 02:55:49.830:      55.55.55.0/24 via 0.0.0.0 in 8 hops
*Jun  4 02:55:49.834:      192.168.12.0/24 via 0.0.0.0 in 7 hops
*Jun  4 02:55:49.834:      192.168.13.0/24 via 0.0.0.0 in 7 hops
*Jun  4 02:55:49.838:      192.168.24.0/2
R12#4 via 0.0.0.0 in 6 hops
*Jun  4 02:55:49.838:      192.168.34.0/24 via 0.0.0.0 in 6 hops
*Jun  4 02:55:49.842:      192.168.45.0/24 via 0.0.0.0 in 5 hops
*Jun  4 02:55:49.846:      192.168.56.0/24 via 0.0.0.0 in 4 hops
*Jun  4 02:55:49.846:      192.168.57.0/24 via 0.0.0.0 in 4 hops
*Jun  4 02:55:49.846:      192.168.68.0/24 via 0.0.0.0 in 3 hops
*Jun  4 02:55:49.846:      192.168.78.0/24 via 0.0.0.0 in 3 hops
*Jun  4 02:55:49.846:      192.168.89.0/24 via 0.0.0.0 in 2 hops
*Jun  4 02:55:49.850:      192.168.109.0/24 via 0.0.0.0 in 1 hops
*Jun  4 02:55:49.850:      192.168.119.0/24 via 0.0.0.0 in 2 hops
R12#
*Jun  4 02:55:51.574: RIP: sending v2 update to 192.168.120.1 via Serial1/0 (192.168.120.2)
*Jun  4 02:55:51.574: RIP: build update entries
*Jun  4 02:55:51.578:   11.11.11.0/24 via 0.0.0.0, metric 2, tag 0
*Jun  4 02:55:51.578:   12.12.12.0/24 via 0.0.0.0, metric 1, tag 0
*Jun  4 02:55:51.578:   55.66.66.0/24 via 0.0.0.0, metric 1, tag 0
*Jun  4 02:55:51.578:   192.168.119.0/24 via 0.0.0.0, metric 2, tag 0
*Jun  4 02:55:51.578:   192.168.121.0/24 via 0.0.0.0, metric 1, tag 0
R12#
*Jun  4 02:55:57.442: RIP: sending v2 update to 192.168.121.1 via Serial1/1 (192.168.121.2)
*Jun  4 02:55:57.442: RIP: build update entries
*Jun  4 02:55:57.442:   10.10.10.0/24 via 0.0.0.0, metric 2, tag 0
*Jun  4 02:55:57.442:   12.12.12.0/24 via 0.0.0.0, metric 1, tag 0
*Jun  4 02:55:57.446:   55.66.66.0/24 via 0.0.0.0, metric 1, tag 0
*Jun  4 02:55:57.446:   192.168.109.0/24 via 0.0.0.0, metric 2, tag 0
*Jun  4 02:55:57.446:   192.168.120.0/24 via 0.0.0.0, metric 1, tag 0
R12#
*Jun  4 02:56:09.678: RIP: received packet with MD5 authentication
*Jun  4 02:56:09.678: RIP: received v2 update from 192.168.121.1 on Serial1/1
*Jun  4 02:56:09.678:      1.1.1.0/24 via 0.0.0.0 in 8 hops
*Jun  4 02:56:09.678:      2.2.2.0/24 via 0.0.0.0 in 7 hops
*Jun  4 02:56:09.682:      3.3.3.0/24 via 0.0.0.0 in 7 hops
*Jun  4 02:56:09.686:      4.4.4.0/24 via 0.0.0.0 in 6 hops
*Jun  4 02:56:09.686:      5.5.5.0/24 via 0.0.0.0 in 5 hops
```

To verify that RIP updates are authenticated and sent as unicast, I used the following debugging command:

debug ip rip

This command enables real-time monitoring of RIP activity, including sending and receiving routing updates.

In the debug output, I saw lines such as:

RIP: received packet with MD5 authentication

This confirms that RIP MD5 authentication is successfully configured and that the router is receiving authenticated updates from its neighbors using the shared key.

I also observed entries like:

RIP: sending v2 update to 192.168.120.1 via Serial1/0 (192.168.120.2)

This confirms that RIP updates are being sent as unicast to specific neighbor IP addresses, rather than multicast (which would appear as 224.0.0.9).

This behavior is expected because I had configured:

passive-interface default

neighbor <neighbor-ip>

By default, RIP calculates metrics based on hop count and supports a maximum of 15 hops. I manipulated the metric by increasing the hop count, so the router prefers an alternate path for routing packets.

To complete the task of making each router prefer the longer route to Router 1’s second loopback (55.55.55.0), I configured a metric offset using an access list and the offset-list command in RIP. Here's what I did:

```
access-list 1 permit 55.55.55.0
router rip
 offset-list 1 in 5 Serial1/0
```

I first created an access-list that matches the specific network 55.55.55.0. This is the second loopback on Router 1.
Then I applied an offset of 5 to incoming RIP updates that are received on the Serial1/0 interface.
So basically, I told the router:
“If you learn about network 55.55.55.0 via Serial1/0, pretend it's 5 hops further away than it actually is.”
Example

```
[R1]------[R2]------[R4]
  \                /
   \              /
    \            /
     ----[R3]-----
```

Router 1 (R1) has a second loopback: 55.55.55.1

This loopback is advertised in RIP as 55.55.55.0/24

Router 4 (R4) can reach that network via two paths:

Path 1: R4 ← R2 ← R1 (shorter)

Path 2: R4 ← R3 ← R1 (longer)

You want R4 to prefer the longer path (through R3), not the default shorter one.
Reference: https://lucidresource.com/labs/rip_advanced_lab.pdf