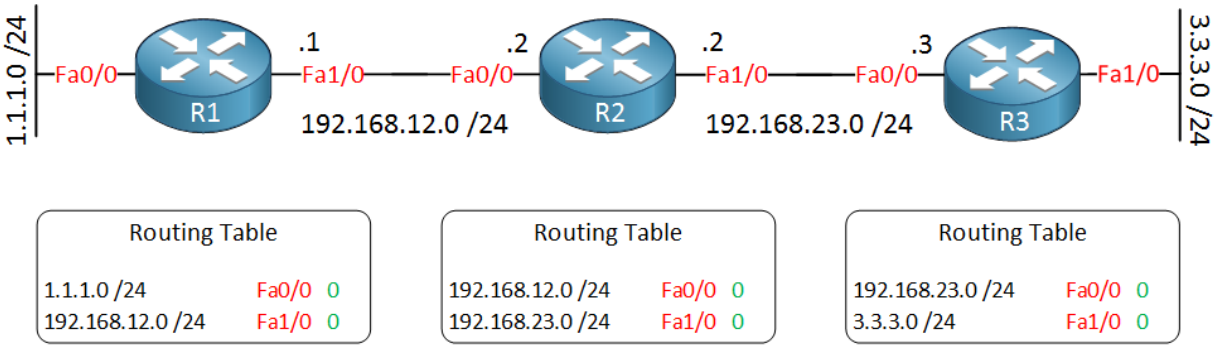
RIP Distance Vector Routing Protocol

RIP is a distance vector routing protocol and the simplest routing protocol to start with. We'll start by paying attention to the distance vector class. What does the name distance vector mean?

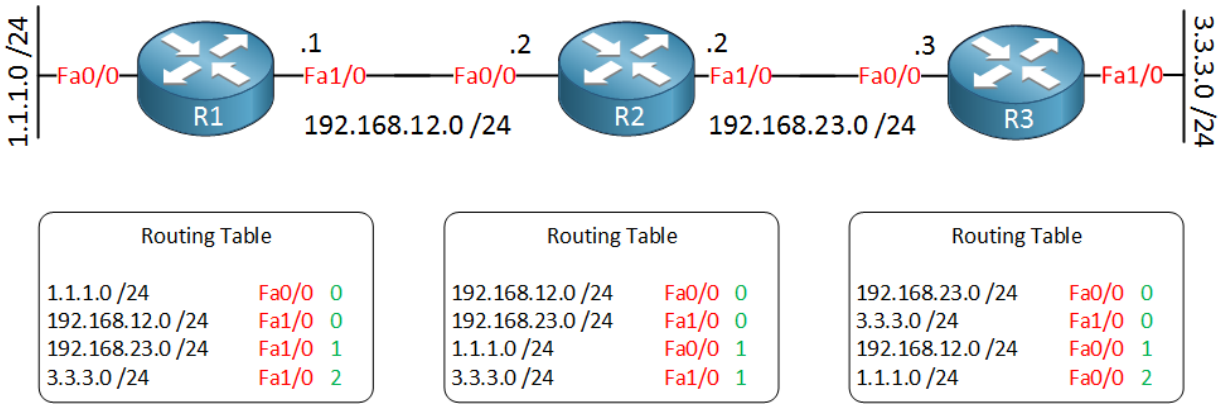
- **Distance**: How far away, in routing world we use metrics which we just discussed.
- **Vector**: Which direction, in routing world we care about which interface and the IP-address of the next router to send it to.



In this picture we have three routers and we are running a distance vector routing protocol (RIP). As we start our routers they build a routing table by default but the only thing they know are their directly connected interfaces. You can see that this information is in their routing table. In **red** you can see which interface and in **green** you can see the metric. RIP uses hop count as its metric which is nothing more than counting the number of routers (hops) you have to pass to get to your destination.

Now I'm going to enable distance vector routing, what will happen is that our routers will **copy their routing table** to their directly connected neighbor. R1 will copy its routing table to R2. R2 will copy its routing table to R3 and the other way around.

If a router receives information about a network it doesn't know about yet, it will add this information to its routing table:



Take a look at R1 and you will see that it has learned about the 192.168.23.0 /24 and 3.3.3.0 /24 network from R2. You see that it has added the interface (Fa1/0) how to reach these networks (that's the vector part) and you see that it has added the metric (hop count) for these networks (that's the distance part).

192.168.23.0 /24 is one hop away, 3.3.3.0 /24 is two hops away.

Awesome! You also see that R2 and R3 have filled their routing tables.

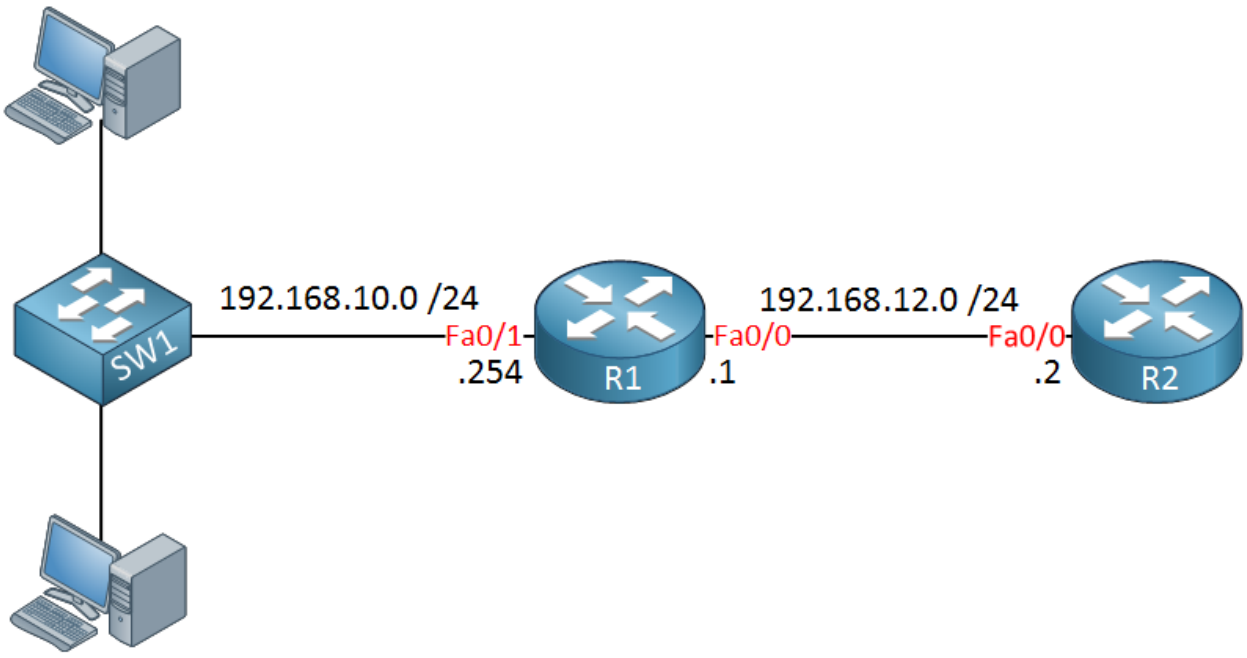
Every 30 seconds our routers will send a full copy of their routing table to their neighbors who can update their own routing table.

RIP Passive Interface

When you use the RIP network command, two things will happen:

- All interfaces that have a network that falls within the range of your network command will be advertised in RIP.
- RIP updates will be sent on these interfaces.

Sometimes, however, you might want to advertise a network in RIP but you don't want to send updates everywhere. Take a look at the topology below for an example:



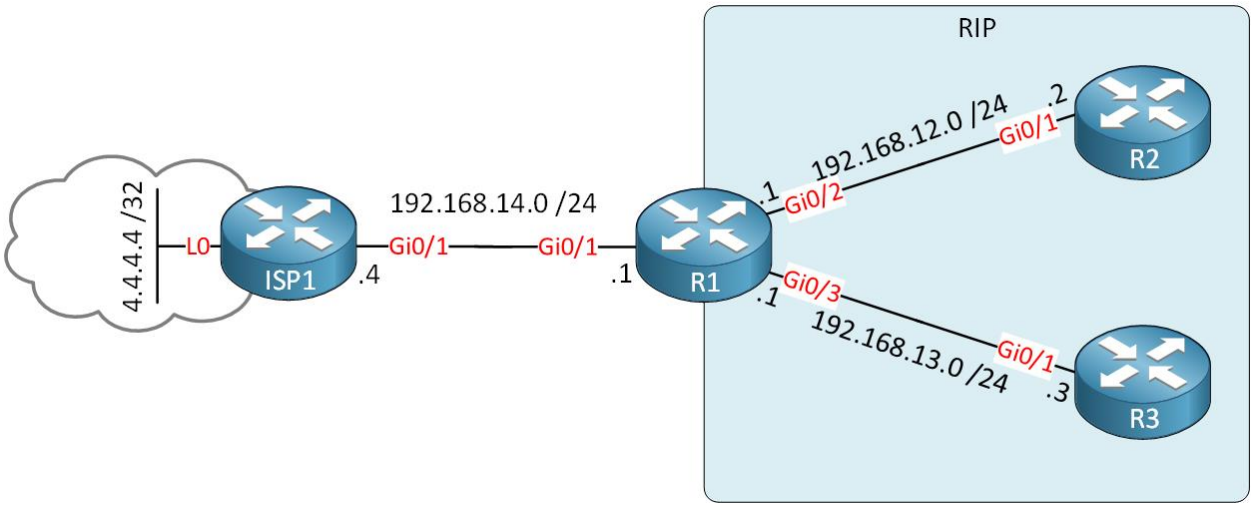
Above we have two routers, R1 and R2. On the left side, there's the 192.168.10.0 /24 network with a switch and some computers. R1 wants to advertise this network to R2 but since there are no other RIP routers in the 192.168.10.0 /24 network, it's pointless to send RIP updates on the FastEthernet 0/1 interface.

To prevent this from happening, we will use the **passive-interface** command. This will ensure that the network is advertised in RIP but it will not send RIP updates on the interface.

bove you can see that the RIP updates are going in both directions.

RIP Default Route

Most routing protocols allow you to advertise a default route, RIP is no exception. This can be useful if you have a single exit point in your network. Let me give you an example:



Above we have a customer network on the right side, using R1, R2, and R3. On the left side, there's the ISP. The loopback on the ISP1 router is supposed to be some network on the Internet.

To reach the Internet, R1, R2, and R3 should have a default route. There are two ways how we can configure this:

- We could configure a static default route on R1, R2 and R3.
- We could configure a static default route on R1 and advertise it in RIP to R2 and R3.
- Let's start with a basic RIP configuration for R1, R2 and R3 so that all internal links are advertised:

```
R1(config)#router rip
R1(config-router)#version 2
R1(config-router)#no auto-summary
R1(config-router)#network 192.168.12.0
R1(config-router)#network 192.168.13.0
R2(config)#router rip
R2(config-router)#version 2
R2(config-router)#no auto-summary
R2(config-router)#network 192.168.12.0
R3(config)#router rip
R3(config-router)#version 2
R3(config-router)#no auto-summary
R3(config-router)#network 192.168.13.0
```

- Let's configure a static default route on R1 to reach the networks behind the ISP1 router:

```
R1(config)#ip route 0.0.0.0 0.0.0.0 192.168.14.4
```

- Let's see if it works:

```
R1#ping 4.4.4.4
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 4.4.4.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/2/4 ms
```

- No problems there. Time to advertise it in RIP:

```
R1(config)#router rip
R1(config-router)#default-information originate
```

- The command above will tell RIP to advertise the static default route.
- Let's see what we have on R2 and R3:

```
R2#show ip route rip

Gateway of last resort is 192.168.12.1 to network 0.0.0.0

R*    0.0.0.0/0 [120/1] via 192.168.12.1, 00:00:26, GigabitEthernet0/1
R     192.168.13.0/24 [120/1] via 192.168.12.1, 00:00:26, GigabitEthernet0/1
R3#show ip route rip

Gateway of last resort is 192.168.13.1 to network 0.0.0.0

R*    0.0.0.0/0 [120/1] via 192.168.13.1, 00:00:18, GigabitEthernet0/1
R     192.168.12.0/24 [120/1] via 192.168.13.1, 00:00:18, GigabitEthernet0/1
```

- Both routers have a default route, learned from R1. Let's test these:

```
R2#ping 4.4.4.4
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 4.4.4.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 3/3/4 ms
R3#ping 4.4.4.4
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 4.4.4.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 3/4/6 ms
```

- Our default route learned through RIP is working.
- The pings from R2 and R3 will only work if you have a route on the ISP1 router for the 192.168.12.0/24 and 192.168.13.0/24 networks. I achieved this with a static route on the ISP1 router.

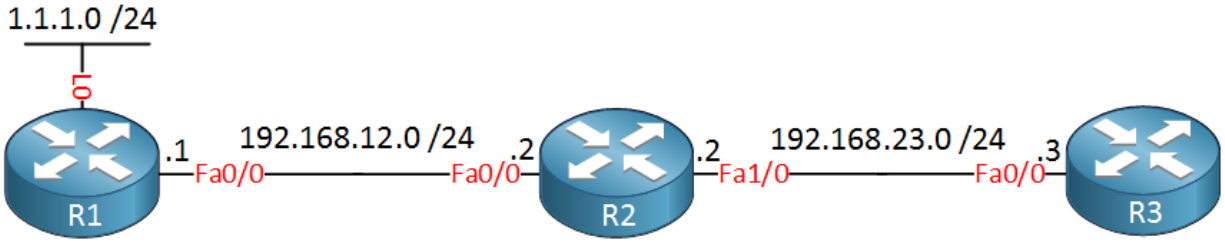
RIP Timers Debug

RIP uses a couple of timers to do its work:

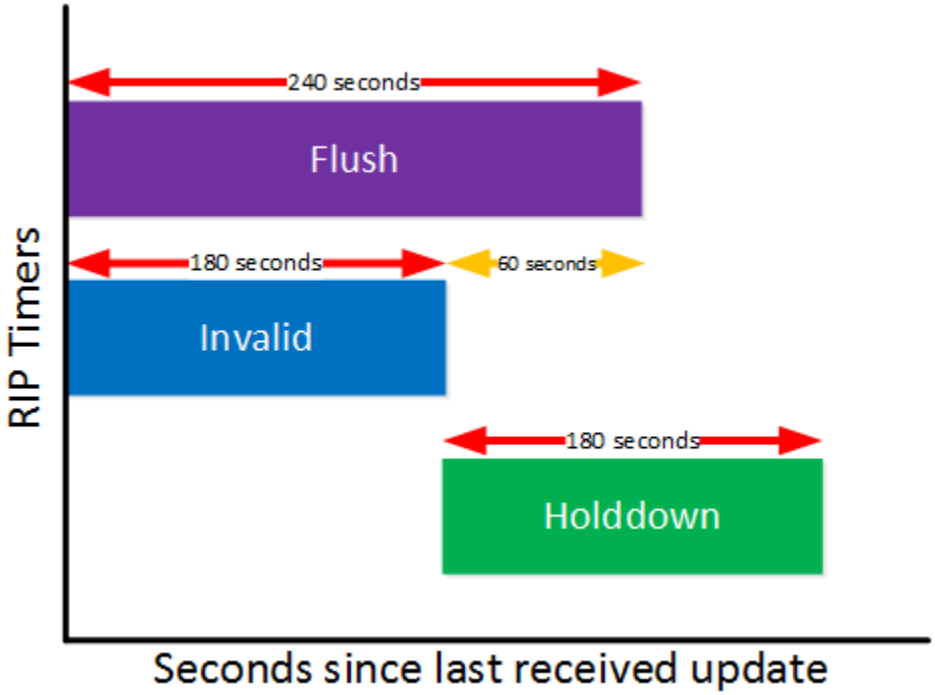
- **Update**: this is how often we send routing updates, the default is 30 seconds.

- **Invalid**: the number of seconds since we received the last valid update, once this timer expires the route goes into holddown, the default is 180 seconds.
- **Holddown**: the number of seconds that we wait before we accept any new updates for the route that is in holddown, the default is 180 seconds,
- **Flush**: how many seconds since we received the last valid update until we throw the route away, the default is 240 seconds.

In this lesson, we'll take a look at these times in action. We will use the following topology for this:



Because of this, the holddown timer is only active for 60 seconds, not 180 seconds. Here's an image to help you visualize this:



Troubleshooting RIP

an old distance vector routing protocol that uses hop count as its metric. Unlike OSPF or EIGRP, RIP doesn't establish a neighbor adjacency. Most of the RIP troubleshooting issues are about missing routing information.

Here are a number of things that could go wrong with RIP:

- **Wrong network command(s):** the network command is used to tell RIP what networks to advertise but also where to send RIP routing updates to. Wrong (or missing) network commands will cause issues.
- **Interface shut**: A network on an interface that is in shutdown will not be advertised.
- **Passive interface:** An interface that has been configured as passive will not send any RIP updates.
- **Version mismatch:** RIP has two versions, both routers should use the same version.
- **Max hop count:** When the hop count is 16, the network is considered unreachable.
- **Route Filtering:** Filters might prevent RIP updates from beint sent or received.
- **Authentication:** Both RIP routers should have the same authentication parameters.
- **Split horizon:** Networks that are learned on an interface are not advertised out of the same interface.
- **Auto-summarization:** Causes issues with discontigious networks.

Introduction to Route Summarization

Route summarization is a method where we create one summary route that represent multiple networks/subnets. It's also called route aggregation or supernetting.

Summarization has a number of advantages:

- **Saves memory**: routing tables will be smaller which reduces memory requirements.
- **Saves bandwidth**: there are less routes to advertise so we save some bandwidth.
- **Saves CPU cycles**: less packets to process and smaller routing tables to work on.
- **Stability**: Prevents routing table instability due to flapping networks.

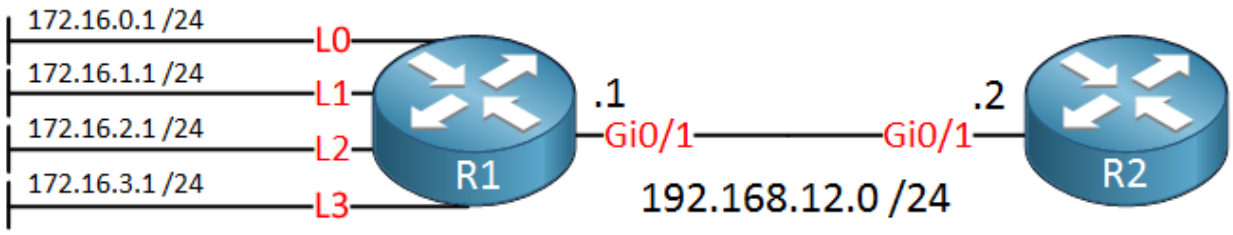
There are also some disadvantages to summarization:

- **Forwarding traffic for unused networks**: a router will drop traffic when it doesn't have a matching destination in its routing table. When we use summarization, it's possible that the summary route covers networks that are not in use. The router that has a summary route will forward them to the router that has advertised the summary route.

- **Sub-optimal routing**: routers prefer the path with the longest prefix match. When you use summaries, it's possible that your router prefers another path where it has learned a more specific network from. The summary route also has a single metric.

Configuration

This is the topology we will use:



Without Route Summarization

Let's configure RIP so that all loopback interfaces are advertised:

```
R1(config)#router rip

R1(config-router)#version 2

R1(config-router)#no auto-summary

R1(config-router)#network 172.16.0.0

R1(config-router)#network 192.168.12.0

R2(config)#router rip

R2(config-router)#version 2

R2(config-router)#no auto-summary

R2(config-router)#network 192.168.12.0
```

Let's enable a debug so that we can see what is going on behind the scenes:

```
R1 & R2
```

```
#debug ip rip

RIP protocol debugging is on
```

Here's what we will see:

```
R1#

RIP: sending v2 update to 224.0.0.9 via GigabitEthernet0/1 (192.168.12.1)

RIP: build update entries

        172.16.0.0/24 via 0.0.0.0, metric 1, tag 0

        172.16.1.0/24 via 0.0.0.0, metric 1, tag 0

        172.16.2.0/24 via 0.0.0.0, metric 1, tag 0

        172.16.3.0/24 via 0.0.0.0, metric 1, tag 0
```

R1 is advertising four different networks. R2 receives them:

```
R2#

RIP: received v2 update from 192.168.12.1 on GigabitEthernet0/1

     172.16.0.0/24 via 0.0.0.0 in 1 hops

     172.16.1.0/24 via 0.0.0.0 in 1 hops

     172.16.2.0/24 via 0.0.0.0 in 1 hops

     172.16.3.0/24 via 0.0.0.0 in 1 hops
```

The more information we advertise, the more bandwidth we require and more CPU cycles are required to process them. Of course, four networks on a Gigabit interface are no problem but in larger networks, there might be thousands or hundred of thousands of networks that are advertised.

Let's check R2:

```
R2#show ip route rip



R     172.16.0.0/24 [120/1] via 192.168.12.1, 00:00:11, GigabitEthernet0/1

R     172.16.1.0/24 [120/1] via 192.168.12.1, 00:00:11, GigabitEthernet0/1

R     172.16.2.0/24 [120/1] via 192.168.12.1, 00:00:11, GigabitEthernet0/1

R     172.16.3.0/24 [120/1] via 192.168.12.1, 00:00:11, GigabitEthernet0/1
```

R2 stores all networks in its routing table which requires memory.

Let's talk about stability. Let me show you what happens when we shut one of the loopback interfaces on R1:

```
R1(config)#interface loopback 0

R1(config-if)#shutdown
```

As soon as this happens, R1 will send a triggered update to R2:

```
RIP: sending v2 flash update to 224.0.0.9 via GigabitEthernet0/1 (192.168.12.1)

RIP: build flash update entries

        172.16.0.0/24 via 0.0.0.0, metric 16, tag 0
```

R2 receives this update:

```
R2#

RIP: received v2 update from 192.168.12.1 on GigabitEthernet0/1

     172.16.0.0/24 via 0.0.0.0 in 16 hops  (inaccessible)
```

After awhile, R2 will remove this network from its routing table. Every time an interface goes up and down, packets are generated and the routing table will change. All of this requires

bandwidth, CPU cycles, and memory. No problem for our small network but when you have thousands of networks and dozens of routers then it's a different story.

With Route Summarization

Let's see how route summarization works. I'll configure R1 to advertise a summary towards R2:

```
R1(config)#interface GigabitEthernet 0/1

R1(config-if)#ip summary-address rip 172.16.0.0 255.255.0.0
```

Here's what R1 advertises now:

```
RIP: sending v2 update to 224.0.0.9 via GigabitEthernet0/1 (192.168.12.1)

RIP: build update entries

        172.16.0.0/16 via 0.0.0.0, metric 1, tag 0
```

And here's what R2 receives:

```
RIP: received v2 update from 192.168.12.1 on GigabitEthernet0/1

     172.16.0.0/16 via 0.0.0.0 in 1 hops
```

Only one network is advertised, our summary route. Less information, less bandwidth, less CPU cycles required and less memory. Here's the routing table of R2:

```
R2#show ip route rip



R     172.16.0.0/16 [120/1] via 192.168.12.1, 00:00:10, GigabitEthernet0/1
```

Only one entry remains. R2 is still able to reach every network that our summary route covers. Let's try this:

```
R2#ping 172.16.1.1
```

```
Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 4/5/6 ms
```

This is looking good. Let me also show you one of the disadvantages of summarization. Here's what happens when we ping an IP address that is covered by the summary route but which is not available:

```
R2#ping 172.16.4.4

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 172.16.4.4, timeout is 2 seconds:

U.U.U

Success rate is 0 percent (0/5)
```

The U means it's unreachable. R2 has a matching route towards R1 so it sends these packets to R1. When R1 receives them, it drops them since it doesn't have a matching entry and informs R2 about this.

Remember what happened when we shut one of the loopback interface? RIP would send a triggered update. Let's try that again:

```
R1(config)#interface loopback 0

R1(config-if)#shutdown
```

Nothing will happen now! As long as there is one interface up with an IP address that falls within the summary route then the summary will be advertised. This makes our network far more stable.

Let me show you what happens when I shut the remaining loopbacks:

```
R1(config)#interface loopback 1
```

```
R1(config-if)#shutdown



R1(config)#interface loopback 2

R1(config-if)#shutdown



R1(config)#interface loopback 3

R1(config-if)#shutdown
```

Once I shut the last loopback, something happens:

```
R1#

RIP: sending v2 flash update to 224.0.0.9 via GigabitEthernet0/1 (192.168.12.1)

RIP: build flash update entries

        172.16.0.0/16 via 0.0.0.0, metric 16, tag 0
```

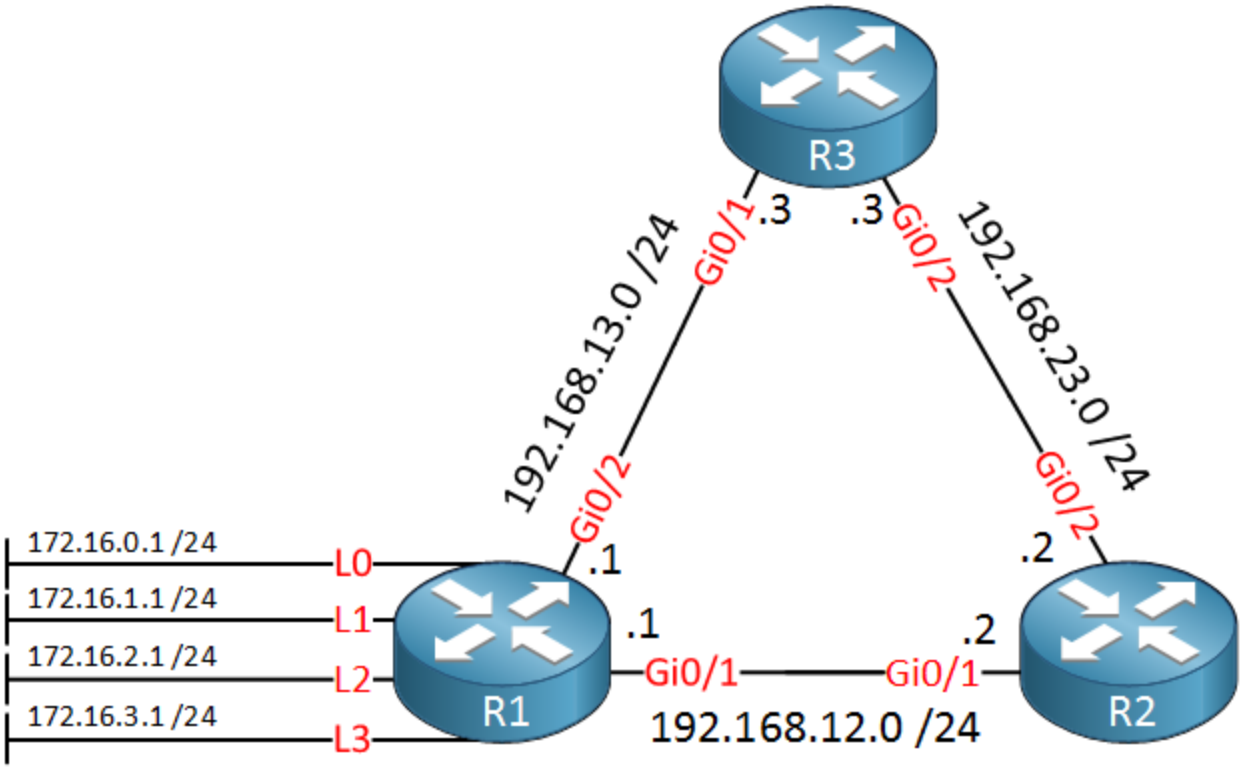
Once the final loopback interface is gone, R1 sends the triggered update to inform R2 that the summary route is gone. Here's R2:

```
R2#

RIP: received v2 update from 192.168.12.1 on GigabitEthernet0/1

     172.16.0.0/16 via 0.0.0.0 in 16 hops  (inaccessible)
```

There is one more disadvantage to summarization but to demonstrate this, I will have to add another router. Let's add R3:



We use the same topology but R1 and R2 are now also connected to R3. Let's make sure our loopback interfaces are up again:

```
R1(config)#interface range loopback 0 - 3

R1(config-if-range)#no shutdown
```

Let's configure R1 so that it sends RIP packets to R3:

```
R1(config)#router rip

R1(config-router)#network 192.168.13.0



R1(config)#access-list 1 deny any



R1(config)#router rip
```

```
R1(config-router)distribute-list 1 in GigabitEthernet 0/2
```

The access-list on R1 is required to prevent R1 from installing any RIP routes from R3. Otherwise the summary route is advertised from R1 > R2 > R3 > R1, causing a routing loop.

Let's configure R2:

```
R2(config-if)#router rip

R2(config-router)#network 192.168.23.0
```

And R3:

```
R3(config-if)#router rip

R3(config-R3)#version 2

R3(config-R3)#no auto-summary

R3(config-R3)#network 192.168.13.0

R3(config-R3)#network 192.168.23.0
```

Now let's take a look at the routing tables:

```
R3#show ip route rip



      172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks

R        172.16.0.0/16 [120/2] via 192.168.23.2, 00:00:22, GigabitEthernet0/2

R        172.16.0.0/24 [120/1] via 192.168.13.1, 00:00:24, GigabitEthernet0/1

R        172.16.1.0/24 [120/1] via 192.168.13.1, 00:00:24, GigabitEthernet0/1

R        172.16.2.0/24 [120/1] via 192.168.13.1, 00:00:24, GigabitEthernet0/1
```

```
R        172.16.3.0/24 [120/1] via 192.168.13.1, 00:00:24, GigabitEthernet0/1

R     192.168.12.0/24 [120/1] via 192.168.23.2, 00:00:22, GigabitEthernet0/2

                      [120/1] via 192.168.13.1, 00:00:24, GigabitEthernet0/1
```

Above you can see that R3 learns the summary route from R2, all other networks are learned from R1. Let's check R2:

```
R2#show ip route rip



      172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks

R        172.16.0.0/16 [120/1] via 192.168.12.1, 00:00:08, GigabitEthernet0/1

R        172.16.0.0/24 [120/2] via 192.168.23.3, 00:00:10, GigabitEthernet0/2

R        172.16.1.0/24 [120/2] via 192.168.23.3, 00:00:10, GigabitEthernet0/2

R        172.16.2.0/24 [120/2] via 192.168.23.3, 00:00:10, GigabitEthernet0/2

R        172.16.3.0/24 [120/2] via 192.168.23.3, 00:00:10, GigabitEthernet0/2

R     192.168.13.0/24 [120/1] via 192.168.23.3, 00:00:10, GigabitEthernet0/2

                      [120/1] via 192.168.12.1, 00:00:08, GigabitEthernet0/1
```

Above you can see that we now have sub-optimal routing on R2. Since the router prefers the most specific path, it will use R3 to reach the four 172.16.x.0/24 networks and it's not using the summary route from R1. We can verify this with a traceroute:

```
R2#traceroute 172.16.0.1 probe 1

Type escape sequence to abort.

Tracing the route to 172.16.0.1

VRF info: (vrf in name/id, vrf out name/id)
```

```
1 192.168.23.3 5 msec

2 192.168.13.1 10 msec
```

There are a couple of different methods how you can solve this, depending on what routing protocol you select.