how to master CCNP ROUTE

René Molenaar

Introduction

One of the things I do in life is work as a Cisco Certified System Instructor (CCSI) and after teaching CCNA/CCNP for a few years I've learned which topics people find difficult to understand. This is the reason I created http://gns3vault.com where I offer free Cisco labs and videos to help people learn networking. The problem with networking is that you need to know what you are doing before you can configure anything. Even if you have all the commands you still need to understand *what* and *why* you are typing these commands. I created this book to give you a compact guide which will provide you the answer to *what* and *why* to help you master the CCNP ROUTE exam.

I have tried to put all the important keywords in **bold**. If you see a **term or concept** in **bold** it's something you should remember / write down and make sure you understand it since its core knowledge for your CCNA!

One last thing before we get started. When I'm teaching I always advise students to create mindmaps instead of notes. Notes are just lists with random information while mindmaps show the relationship between the different items. If you are reading this book on your computer I highly suggest you download "Xmind" which you can get for free here:

http://xmind.net

If you are new to mindmapping, check out "Appendix A – How to create mindmaps" at the end of this book where I show you how I do it.

I also highly recommend you to follow me along when I'm demonstrating the configuration examples. Boot up GNS3 and configure the examples I'm showing you by yourself. You'll learn more by *actively* working on the routers compared to just *passive* reading.

Enjoy reading my book and good luck getting your CCNP ROUTE certification!

René Molenaar

P.S. If you have any questions or comments about this book, please let me know:

E-mail: info@gns3vault.com
Website: gns3vault.com
Facebook: facebook.com/gns3vault
Twitter: twitter.com/gns3vault
Youtube: youtube.com/gns3vault

Index

1. Introduction to EIGRP

The first routing protocol we will look at is called **EIGRP** (Enhanced Interior Gateway Routing Protocol). EIGRP was created by Cisco which means you can only run it on Cisco hardware. If you want routing with devices from different vendors (like Juniper) you will have to look for another routing protocol.

In this chapter I'm going to give you an introduction to EIGRP, we'll see how it works and how EIGRP is different compared to OSPF. Most of the information in this chapter is a review of EIGRP on CCNA level so if you have everything still fresh in mind you might want to skim through the chapter. Let me start by giving you an overview:

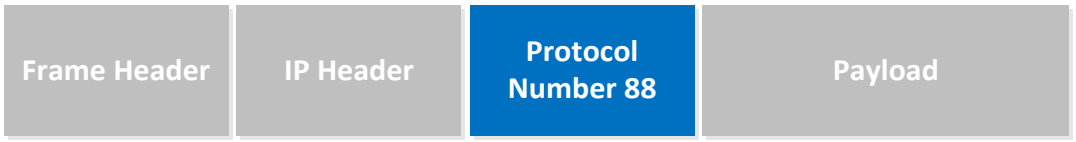
- **Advanced distance vector** or **Hybrid routing protocol**.
- Multicast or unicast is used for exchange of information.
- Multiple network layer protocols are supported.
- 100% loop-free.

Why do we call EIGRP an advanced distance vector or hybrid routing protocol? If you studied CCNA you have seen RIP. RIP is a true distance vector routing protocol and very simple:

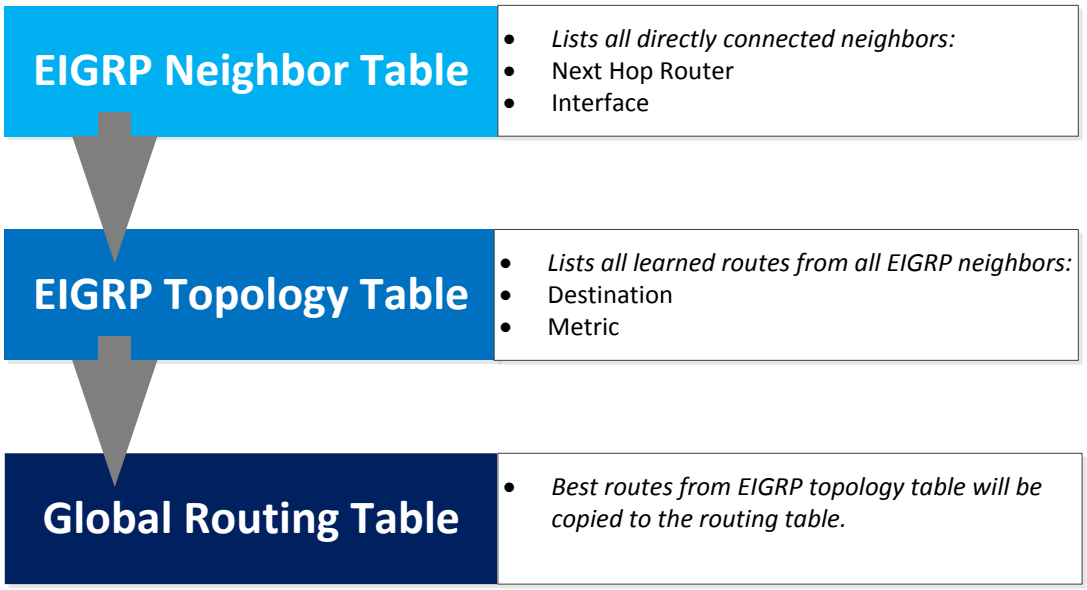
- No neighbor discovery.
- Periodic updates.
- Vulnerable to loops.
- Simple metric (hop count).

Cisco added some of the features from link-state routing protocols to EIGRP which makes it far more advanced than a true distance vector routing protocol like RIP. This is why (probably the marketing department) calls EIGRP an advanced distance vector or hybrid routing protocol.

EIGRP does not use broadcast packets to send information to other neighbors but will use multicast or unicast. Besides IPv4 you can also use EIGRP to route IPv6 or even some older network layer protocols like IPX or AppleTalk. Last but not least…EIGRP is 100% loop-free and I'm going to show you why this is true.

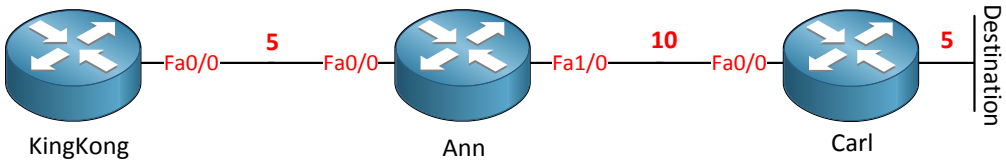


EIGRP runs directly on top of the IP header. If you look at the picture above you see we have a frame header (for example an Ethernet Frame), an IP Header (we are using IPv4) and inside the IP packet you'll find EIGRP. EIGRP has its own protocol number which is 88. Other protocol numbers you are familiar with are TCP (6) and UDP (17).



EIGRP routers will start sending hello packets to other routers just like OSPF does, if you send hello packets and you receive them you will become neighbors. EIGRP neighbors will exchange routing information which will be saved in the topology table. The best path from the topology table will be copied in the routing table.

Selecting the best path with EIGRP works a bit different than other routing protocols so let's see it in action:

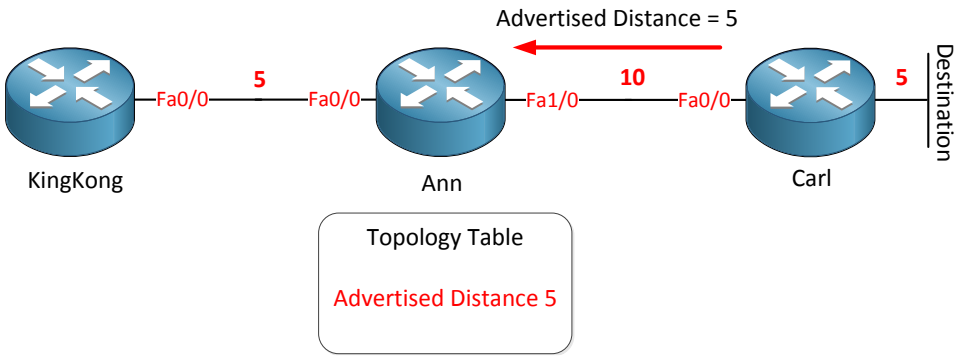


We have three routers named KingKong, Ann and Carl. We are going to calculate the best path to the destination which is behind router Carl.

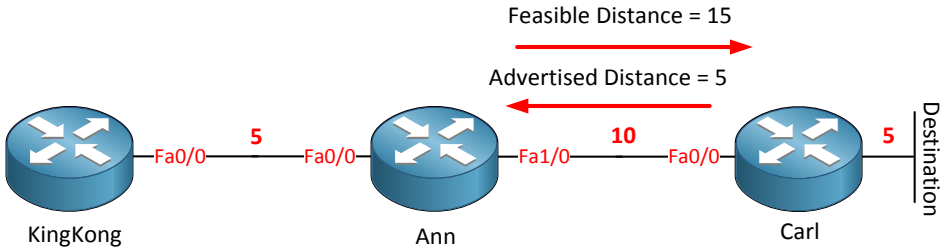
EIGRP uses a rich set of metrics namely **bandwidth, delay, load and reliability** which we will cover later. These values will be put into a formula and each link will be assigned a metric. The lower these metrics the better.

In the picture above I have assigned some values on the interfaces, if you would look on a real EIGRP router you'll see the numbers are very high and a bit annoying to work with.

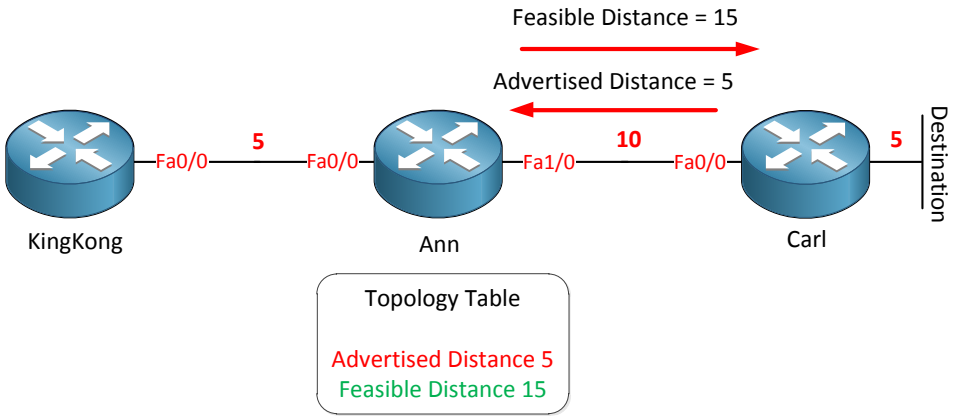
Router Carl will advertise to router Ann its metric towards the destination. Basically router Carl is saying to router Ann: "It costs me 5 to get there". This is called the **advertised distance.**



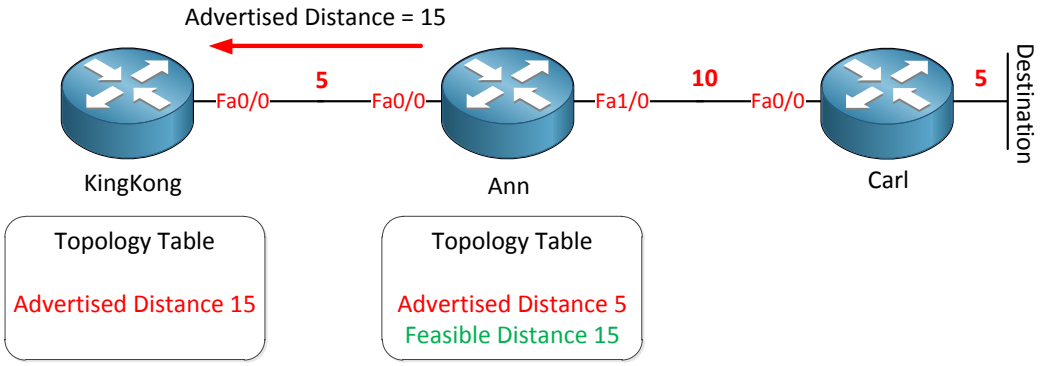
Router Ann has a topology table and in this topology table it will save this metric, the advertised distance to reach this destination is 5.



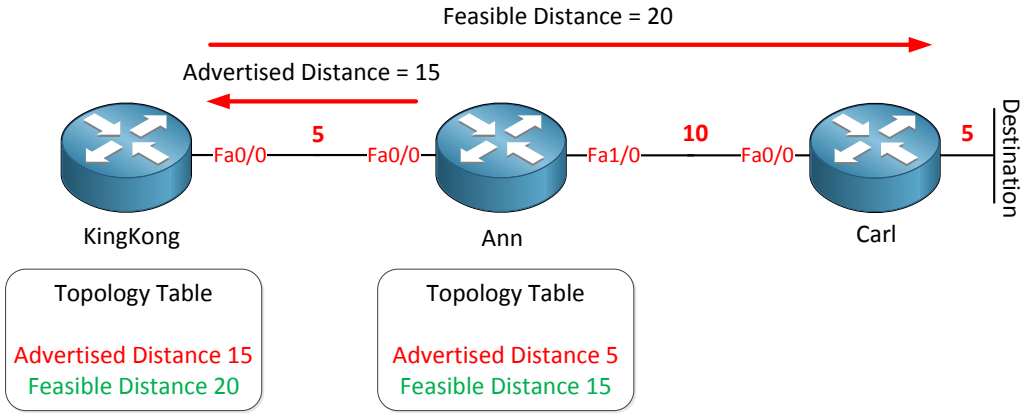
We are not done yet since there is something else that router Ann will save in its topology table. We know the advertised distance is 5 since this is what router Carl told us. We also know the metric of the link between router Ann and router Carl since this is directly connected. Router Ann now knows the metric for the total path to the destination, this total path is called the **feasible distance** and it will be saved in the topology table.



You have now learned two important concepts of EIGRP. The advertised distance, your neighbor tells you how far it is for him to reach the destination and the feasible distance which is your total distance to get to the destination.



We are not done yet since router KingKong is also running EIGRP. Router Ann is sending its feasible distance towards router KingKong which is 15. Router KingKong will save this information in the topology table as the advertised distance. Router Ann is "telling" router KingKong the distance is 15.



Router KingKong now knows how far the destination is away for Router Ann and since we know the metric for the link between router KingKong and Ann it can also calculate the total distance which is called the feasible distance. This information is saved in the topology table.

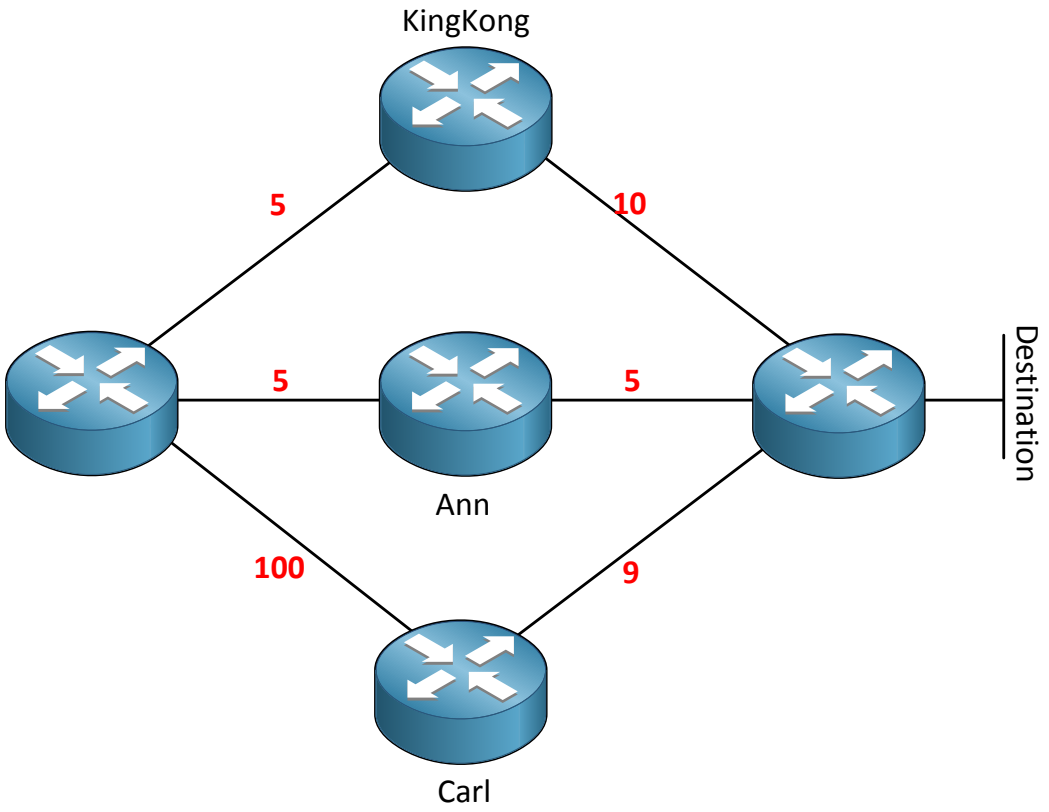
Are you following me so far? Let me describe these terms once again but in plain English:

- ✓ Advertised distance: How far the destination is away for your neighbor.
- ✓ Feasible distance: The total distance to the destination.

The best path to the destination is called the **successor**!

The successor will be copied from the topology table to the routing table.

With EIGRP however it's possible to have a backup path which we call the **feasible successor**. How do we find out if we have a feasible successor? Let's find out:



In the example above we have a couple of routers running EIGRP; we are sitting on the router without a name on the left side and would like to know two things:

- ✓ Which path is the successor (the best path)?
- ✓ Do we have any feasible successors? (backup paths)

Let's fill in the following table to find out:

	Advertised Distance	Feasible distance	
KingKong			
Ann			
Carl			

If you want to try your new-learned EIGRP skills try to fill in the advertised and feasible distance by yourself in the table above.

Router KingKong is telling us the destination is 10 away, router Ann tells us its 5 away and router Carl tells us its 9 away. We can now fill in the advertised distance part of the table:

	Advertised Distance	Feasible distance	
KingKong	10		
Ann	5		
Carl	9		

Since we know our directly connected links we can add this to the advertised distance and we'll have our feasible distance.

	Advertised Distance	Feasible distance	
KingKong	10	15	
Ann	5	10	
Carl	9	109	

The path with the lowest feasible distance will be the successor (router Ann) so now we answered the first question.

	Advertised Distance	Feasible distance	
KingKong	10	15	
Ann	5	10	SUCCESSOR
Carl	9	109	

You will find the successor in the routing table.

To answer the second question "do we have a feasible successor (backup path)?" we need to learn another formula:

Advertised distance of feasible successor < Feasible distance of successor.

This is where I get to see glazed eyes and flabbergasted students so let's do it in plain English one more time:

A router can become a backup path if he is closer to the destination than the total distance of your best path.

I think that sounds a bit better right? Let's try it and see if router KingKong or router Carl is suitable as a backup path:

The advertised distance of router KingKong is 10 which is equal to the feasible distance of router Ann which is also 10. It has to be lower…equal is not good enough so router KingKong will NOT be a feasible successor.

The advertised distance of router Carl is 9 which is lower than the feasible distance of router Ann which is 10. Router Carl will be a valid feasible successor and used as a backup path!

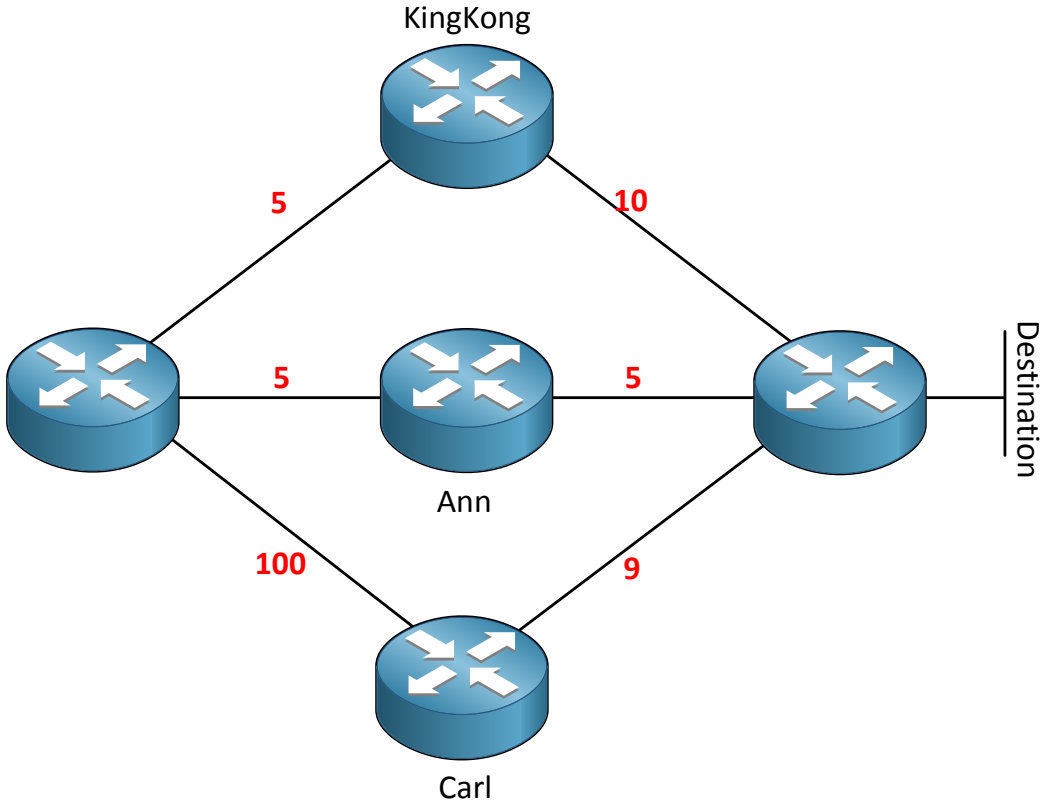
	Advertised Distance	Feasible distance	
KingKong	10	15	
Ann	5	10	SUCCESSOR
Carl	9	109	FEASIBLE SUCCESSOR

Excellent so router Ann is our successor and router Carl is a feasible successor. You will find both entries in the EIGRP topology but you will only find the successor in the routing table. If you lose the successor because of a link failure EIGRP will copy/paste the feasible successor in the routing table. This is what makes EIGRP a FAST routing protocol…but only if you have feasible successor in the routing table.

Now look closely to the feasible distance of router Carl and router KingKong…what do you see? The metric for router Carl is FAR worse than the one for router KingKong. Does this make any sense? Did the Cisco EIGRP engineers make a horrible mistake here by using non-optimal backup paths?

Nope this is perfectly the way it should be! Keep in mind EIGRP at heart is a distance vector protocol. It doesn't know what the complete network looks like…it's not a link-state routing protocol like OSPF which DOES have a complete map of the network. Distance vector routing protocols only know which way to go (vector) and how far away the destination is (distance). I'll show you in a bit exactly why EIGRP works like this.

EIGRP has another trick in its hat. RIP and OSPF both can do load balancing but the paths have to be equal. EIGRP can do something cool…unequal load balancing! Even better it will share traffic in a proportional way, if you have a feasible successor that has a feasible distance which is 5 times worse than the successor traffic will be shared in a 5:1 way.



	Advertised Distance	Feasible distance	
KingKong	10	15	
Ann	5	10	SUCCESSOR
Carl	9	109	FEASIBLE SUCCESSOR

This is our first example where we found out the successor and feasible successor. If you look at the routing table you will only find the successor there. Now we are going to change things so we'll see the feasible successor in the routing table as well so it will load-balance.

You can do this by using the **variance** command. The variance command works as a multiplier:

- ✓ Our successor has a feasible distance of 10.
- ✓ Our feasible successor has a feasible distance of 109.

In order to load-balance our feasible successor needs to have a lower feasible distance than the successor X multiplier.

If we set the variance at 2, this is what we get:

Feasible distance of successor is 10 x 2 (multiplier) = 20.

109 is higher than 20 so we don't do any load balancing.

If we set the variance at 5, this is what we get:

Feasible distance of successor is 10 x 5 (multiplier) = 50.

109 is still higher than 50 so still no load balancing here.

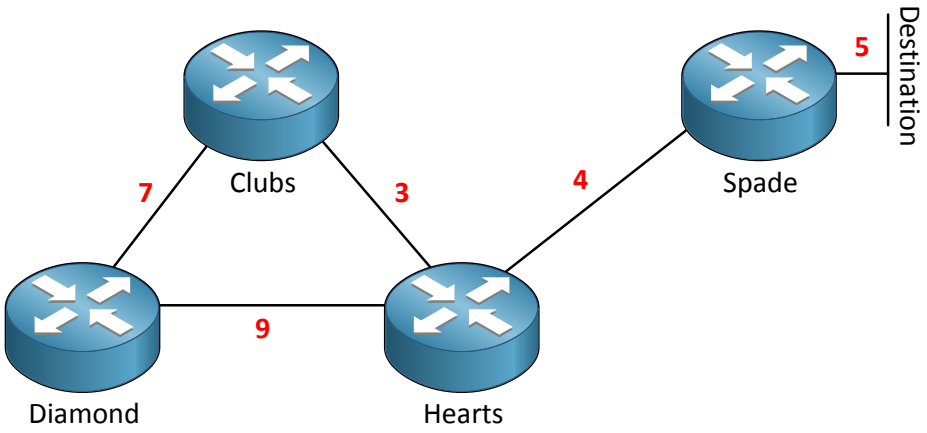
Now I'm going to set the variance at 11 and this is what we get:

Feasible distance of successor is 10 x 11 = 110.

109 is lower than 110 so now we will put the feasible successor in the routing table and start load balancing!

Are we ever going to use the route through router KingKong? No we won't since it's not a feasible successor!

The formula you just witnessed to determine EIGRP feasible successors is how EIGRP can guarantee you that the backup path is 100% loop-free! I know this is difficult to grasp by reading text so let's do another example:



Keep in mind that EIGRP is by nature a distance vector routing protocol, we see the topology but EIGRP does not!

We are looking at the EIGRP topology table of router Hearts and we want to reach the destination behind router Spade, let's fill in the table (try it yourself if you want a good exercise):

	Advertised Distance	Feasible distance	
Spade			
Clubs			
Diamond			

1. Router Spade will advertise the destination network to router Hearts.
2. Router Hearts will advertise the network to router Clubs and Diamond.
3. Router Clubs will advertise the network to router Diamond.
4. Router Diamond will advertise the network to router Clubs.
5. Router Clubs will advertise this network back to router Hearts.
6. Router Diamond will advertise this network back to router Hearts.

	Advertised Distance	Feasible distance	
Spade	5		
Clubs	25		
Diamond	19		

Here we have the advertised distance; our neighbors are telling us how far it is for them to reach the destination network. Next step is to fill in the feasible successors.

How did I get the numbers in the advertised distance table? Let's look at all the routers:

Router Spade is easy. The destination has a distance of 5 as seen in the topology picture. This will be advertised to router Hearts and placed in its topology table.

Router Clubs will learn the destination network through router Hearts and router Diamond. Router Hearts will advertise a distance of 5+4 = 9 to router Clubs. So why didn't I place "9" in the advertised distance field in the table? Good question! Remember split-horizon? Don't advertise to your neighbor whatever you learned from them….router Clubs is not sending information about this network back to router Hearts. To be more specific: **whatever you learn on an interface you don't advertise back out of the same interface.**

How did I get to 25? Let's break it down:

Router Spade will advertise a distance of 5 towards router Hearts. Router Hearts will advertise 5+4 = 9 towards router Diamond.

Router Diamond will advertise 5+4+9 = 18 towards router Clubs. Finally router Clubs will advertise 5+4+9+7 = 25 towards router Hearts. Split-horizon doesn't apply here since router Clubs learned about the destination on another interface (router Diamond).

The same thing applies for the advertised distance of 19 for router Diamond:

1. Router Spade advertises a distance of 5 to router Hearts.
2. Router Hearts advertises a distance of 5+4 = 9 to router Clubs.
3. Router Clubs advertises a distance of 5+4+3 = 12 to router Diamond.
4. Router Diamond advertises a distance of 5+4+3+7 = 19 to router Hearts.

	Advertised Distance	Feasible distance	
Spade	5	9	
Clubs	25	28	
Diamond	19	28	

Router Hearts has learned the advertised distance from its neighbors and knows about its own directly connected interfaces so you can fill in the feasible distance. Last step is to pick our successor.

	Advertised Distance	Feasible distance	
Spade	5	9	SUCCESSOR
Clubs	25	28	
Diamond	19	28	

Router Spade has the lowest feasible distance so it will become the successor…excellent! Let's do the feasible successor check and see if there is a backup path:

Advertised distance of feasible successor < Feasible distance of successor.

The advertised distance of router Clubs (25) and Diamond (19) are higher than the feasible distance of router Spade (9) so they won't become feasible successors. This makes sense right? If these routers become backup paths we would have a loop!

If your neighbor is closer to the destination than your total path you at least know it's not getting to the destination by sending packets **through your router**. Perhaps it's not the best path but it's absolutely 100% loop-free!

This is the end of the EIGRP introduction chapter! What do you think? Was this new for you or just CCNA refreshment? Make sure you understand all the key concepts because in the next chapter we are going to dive deeper into the material.

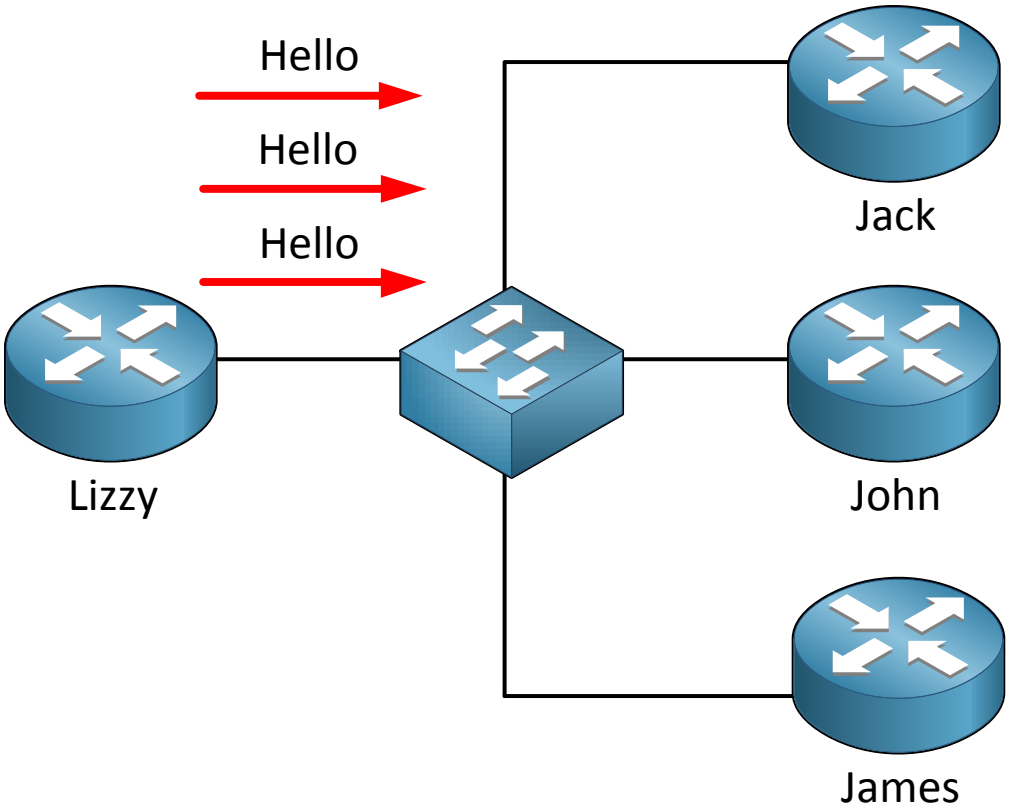
If you want to warm up you might want to try one of my CCNA EIGRP labs that teaches you most of the EIGRP stuff on CCNA level:

http://gns3vault.com/CCNA/eigrp-for-ccna-1.html

2. EIGRP Packets and Metrics

Hello packets are sent between EIGRP neighbors for **neighbor discovery** and **recovery.** If you send hello packets and receive them then EIGRP will form a neighbor relationship with the other router. As long as you receive hello packets from the other side EIGRP will believe that the other router is still there, as soon as you don't receive them anymore you will drop the neighbor relationship called **adjacency** and EIGRP might have to look for another path for certain destinations.

EIGRP uses **RTP (Reliable Transport Protocol)** and its function is to deliver EIGRP packets between neighbors in a **reliable** and ordered way. It can use multicast or unicast and to keep things efficient not all packets are sent reliable. Reliable means that when we send a packet we want to get an **acknowledgment** from the other side to make sure that they received it.



In this example we have 4 routers all running EIGRP. Hello packets are sent between routers in order to form adjacencies. As you can see router Lizzy is sending 3 hello packets meant for router Jack, John and Lizzy.

There are 2 questions that we can ask ourselves here:

- Is it really useful to send 3 different hello packets on a single link?
- Is it necessary that a hello packet gets an acknowledgement in return?

Sending 3 packets on the same link is not very useful so instead of doing this EIGRP will send hello packets by using multicast on a **multi-access** network like Ethernet.

Hello packets don't have to be acknowledged since EIGRP uses a **holddown time**. If a router doesn't receive hello packets in an X amount of time it will drop the neighbor adjacency.
So which packets should be acknowledged? Think about routing information, if there's a change in the network you want to make sure all routers receive this routing update.

Let me show you all the different EIGRP packets:

- **Hello**
- **Update**
- **Query**
- **Reply**
- **ACK** (Acknowledgement)

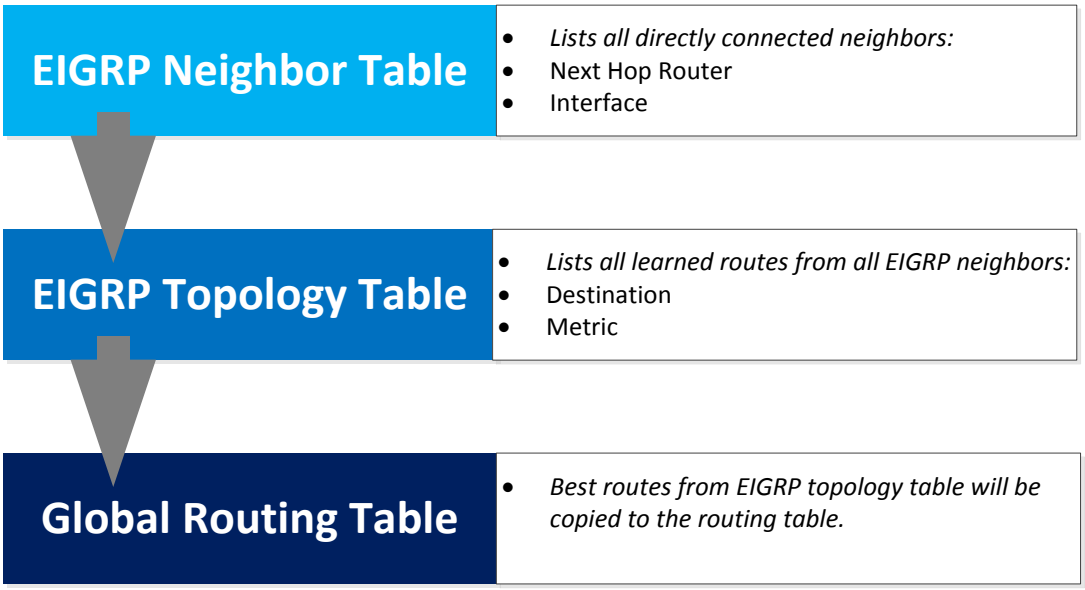
Hello packets are used for neighbor discovery. As soon as you send hello packets and receive them your EIGRP routers will try to form the neighbor adjacency.

Update packets have routing information and are sent reliable to whatever router that requires this information. Update packets can be sent to a single neighbor using unicast or to a group of neighbors using multicast.

Query packets are used when your EIGRP router has lost information about a certain network and doesn't have any backup paths. What happens is that your router will send query packets to its neighbors asking them if they have information about this particular network.

Reply packets are used in response to the query packets and are reliable.

ACK packets are used to acknowledge the receipt of update, query and replay packets. ACK packets are sent by using unicast.

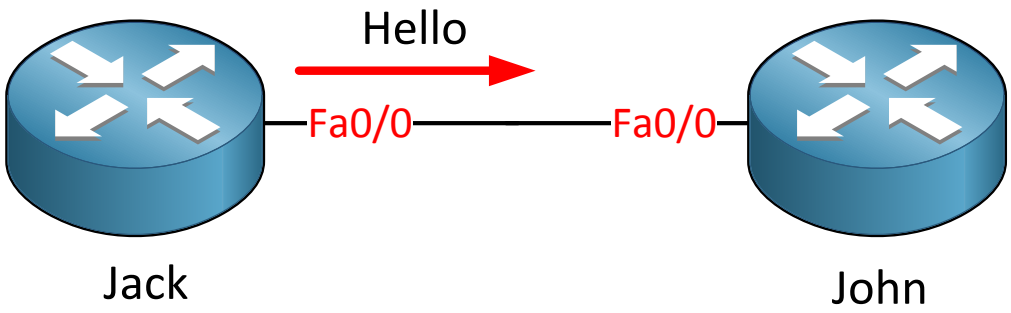


Instead of using just a single routing table EIGRP will use multiple tables. The first one is the **neighbor table** and this is where EIGRP stores all information of directly connected

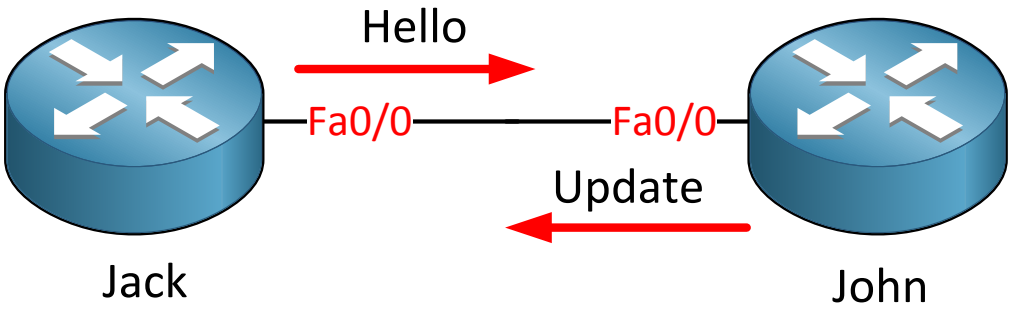
neighbors. After we have become neighbors routers will exchange routing information which is stored in the **EIGRP topology table.** It's possible to have multiple entries for a network in the topology table.

The best information will be copied from the EIGRP topology table to the routing table.

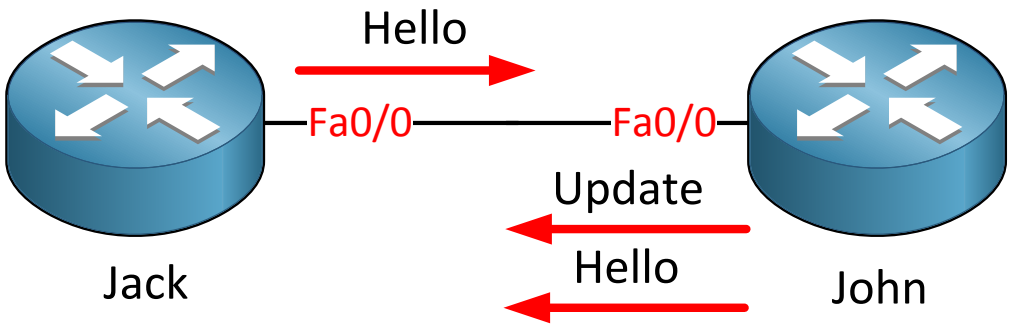
Now you know about all the different packets and the EIGRP tables let's have a look at the total process of becoming EIGRP neighbors and exchanging routing information:



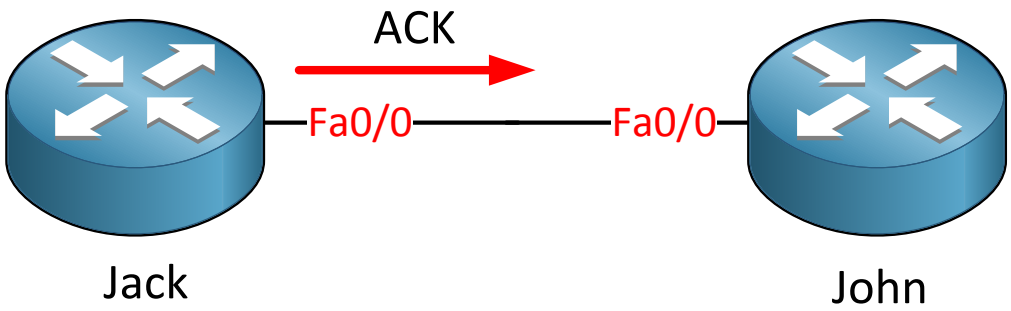
1. We have 2 routers called Jack and John and they are configured for EIGRP. As soon as we enable it for the interface they will start sending hello packets. In this example router Jack is the first router to send a hello packet.



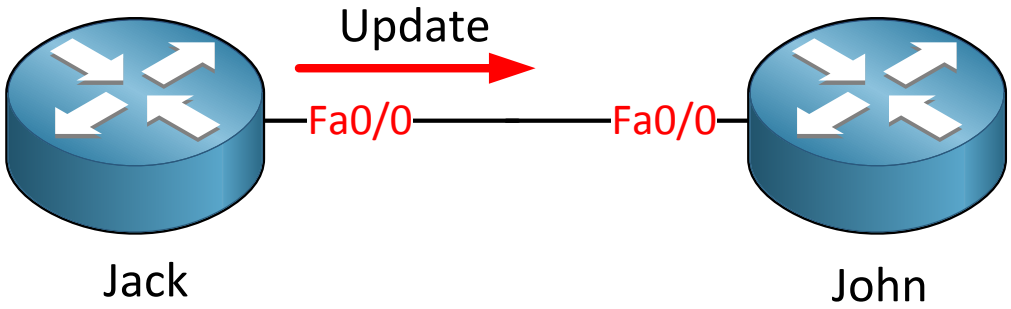
2. As soon as router John receives the hello packet from Jack it will respond by sending update packets that contain all the routing information that it has in its routing table. The only routes that are not sent on this interface are the one that John learned on this interface because of split-horizon. The update packet that router John will send has the initialization bit set so we know this is the "initialization process". At this moment there is still no neighbor adjacency until router John has sent a hello packet to Jack.



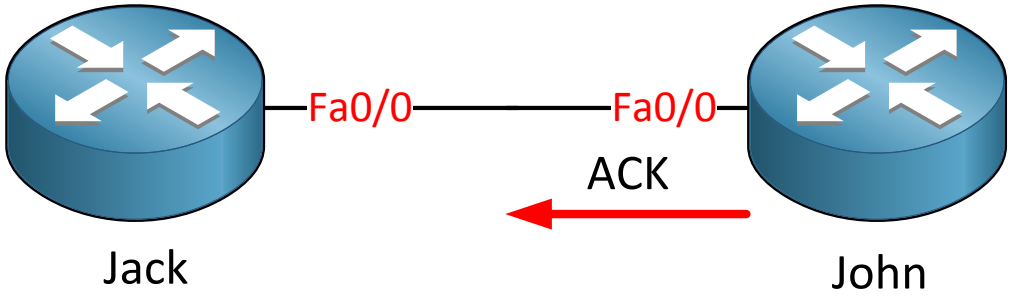
3. Router Jack is of course not the only one sending hello packets. As soon as router John sends a hello packet to Jack we can continue to setup a neighbor adjacency.



4. After both routers have exchanged hello packets we will establish the neighbor adjacency. Router Jack will send an ACK to let John know he received the update packets. The routing information in the update packets will be saved in the EIGRP topology table.



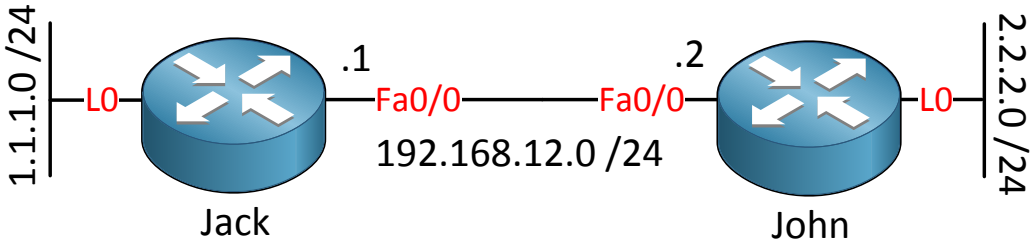
5. Router John is anxious to receive routing information as well so Jack will send update packets to John who will save this information in its EIGRP topology table.



6. After receiving the update packets router John will send an ACK back to Jack to let him know everything is ok.

As soon as both routers have exchanged routing information they will select the best paths to each destination and copy those to the routing table. The best path in EIGRP is called the **successor.**

Want to see what this looks like on a real router? Let's use the following topology and see what happens:



This is the topology I'm going to use to configure EIGRP. My goal is to have full connectivity and here are the configurations:

```
Jack(config)#router eigrp 1
Jack(config-router)#no auto-summary
Jack(config-router)#network 1.1.1.0 0.0.0.255
Jack(config-router)#network 192.168.12.0
Jack(config-router)#exit
```

```
John(config)#router eigrp 1
John(config-router)#no auto-summary
John(config-router)#network 2.2.2.0 0.0.0.255
John(config-router)#network 192.168.12.0
John(config-router)#exit
```

Let's break this one down. **Router eigrp 1** will start up EIGRP using AS (autonomous system) number 1. This number has to **match on both routers** or we won't become EIGRP neighbors.

No auto-summary is needed because by default EIGRP will behave like a classful routing protocol which means it won't advertise the subnet mask along the routing information. In this case that means that 1.1.1.0/24 and 2.2.2.0/24 will be advertised as 1.0.0.0/8 and 2.0.0.0/8. Disabling auto-summary will ensure EIGRP sends the subnet mask along.

Network 1.1.1.0 0.0.0.255 means that I'm advertising networks that exist on interfaces that fall within the 1.1.1.0 – 1.1.1.255 range. If I don't specify the wildcard you'll find "network 1.0.0.0" in your configuration. Does it matter? Yes and no. The same thing applies to "network 2.2.2.0 /24". It will work but also means that every interface that falls within the 1.0.0.0/8 or 2.0.0.0/8 range is going to run EIGRP. Network 192.168.12.0 without a wildcard mask is fine since I'm using a /24 on this interface which is Class C.

If you are working on a lab and are lazy (like me) you can also type in network 0.0.0.0 which will activate EIGRP on all of your interfaces…if that's what you want of course.

Let's do a debug on router John to see what is going on:

```
John#debug eigrp packets ?
  SIAquery  EIGRP SIA-Query packets
  SIAreply  EIGRP SIA-Reply packets
  ack       EIGRP ack packets
  hello     EIGRP hello packets
  ipxsap    EIGRP ipxsap packets
  probe     EIGRP probe packets
  query     EIGRP query packets
  reply     EIGRP reply packets
  request   EIGRP request packets
  retry     EIGRP retransmissions
  stub      EIGRP stub packets
  terse     Display all EIGRP packets except Hellos
  update    EIGRP update packets
  verbose   Display all EIGRP packets
  <cr>
```

As you can see we have a LOT of debug options for EIRP. I want to see the hello packets…

```
John#debug eigrp packets hello
EIGRP Packets debugging is on
    (HELLO)
```

```
John# EIGRP: Received HELLO on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
```

Looking good seems we have received a hello packet from router Jack.

```
John# EIGRP: Sending HELLO on FastEthernet0/0
  AS 1, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
```

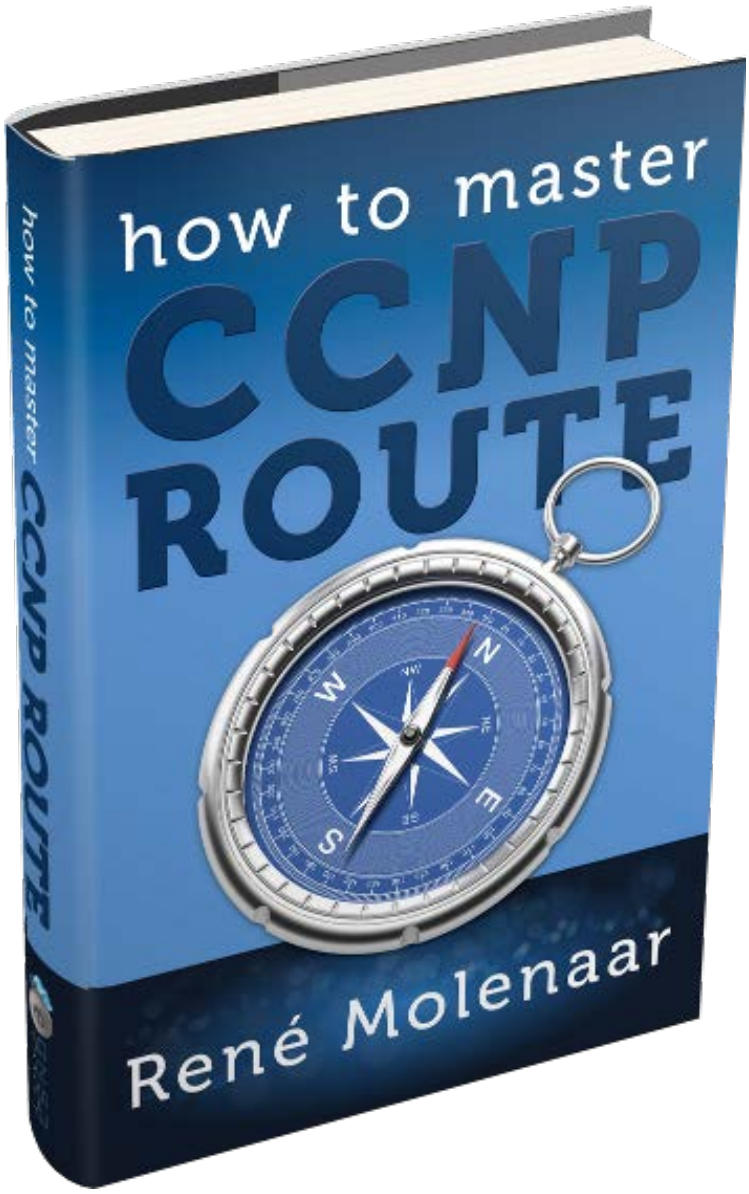
And we are sending hello packets to router Jack as well.

```
Jack# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 192.168.12.2
(FastEthernet0/0) is up: new adjacency
```

Do you enjoy reading this sample of How to Master CCNP ROUTE ?

Click on the link below to get the full version.

Get How to Master CCNP ROUTE Today



```
John# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 192.168.12.1
(FastEthernet0/0) is up: new adjacency
```

You can see we have an EIGRP neighbor adjacency.

```
John# EIGRP: Sending HELLO on Loopback0
AS 1, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0

      EIGRP: Received HELLO on Loopback0 nbr 2.2.2.2
AS 1, Flags 0x0, Seq 0/0 idbQ 0/0
```

Hmm interesting it seems router John is schizophrenic and sending hello packets to its loopback0 interface and also receiving them.

This behavior is normal because the network command does two things:

- Send EIGRP packets on the interface that falls within the network command range.
- Advertise the network that is configured on the interface in EIGRP.

So what do you have to do when you want to advertise a network without sending EIGRP packets on the interface and forming EIGRP neighbors? Use the **passive interface** command.

```
John(config)#router eigrp 1
John(config-router)#passive-interface loopback 0
```

This will advertise the 2.2.2.0/24 network on router John's loopback 0 interface without sending EIGRP packets to the loopback.

```
John(config)#router eigrp 1
John(config-router)#passive-interface default
John(config-router)#no passive-interface fastEthernet 0/0
```

If you have to configure an ISP router with 50+ interfaces you probably don't want to type in this command on each of those interfaces. You can also configure **passive-interface default** and only activate the interfaces you want to run EIGRP on.

Let's look at a debug of some EIGRP update packets. I'm going to clear the EIGRP neighbor adjacency to see what it looks like:

```
John#debug eigrp packets update
EIGRP Packets debugging is on
    (UPDATE)
```

```
Jack#clear ip eigrp neighbors
```

Let's reset the EIGRP neighbor adjacency on router Jack.

```
John# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 192.168.12.1
(FastEthernet0/0) is down: Interface Goodbye received
```

You can see that router Jack is being nice and at least telling router John that it's deleting the neighbor adjacency.

```
John# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 192.168.12.1
(FastEthernet0/0) is up: new adjacency
```

Router Jack is back in business and our EIGRP neighbor adjacency has been reestablished.

```
John#
EIGRP: Enqueueing UPDATE on FastEthernet0/0 nbr 192.168.12.1 iidbQ un/rely
0/1 peerQ un/rely 0/0
EIGRP: Received UPDATE on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x1, Seq 13/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Sending UPDATE on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x1, Seq 13/13 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
```

Here you can see router John is putting an update packet in its queue for router Jack. John is also receiving an update packet from router Jack and afterwards sending its own update packet. You'll see a couple of update packets flying back and forth.

```
John#debug eigrp packets ack
EIGRP Packets debugging is on
    (ACK)
```

We can also check out some acknowledgments by debugging ack packets.

```
Jack#clear ip eigrp neighbors
```

We'll reset the EIGRP neighbor adjacency so the update packets are resent.

```
John#
EIGRP: Received ACK on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/17 idbQ 0/0 iidbQ un/rely 0/1 peerQ un/rely 0/1
EIGRP: Received ACK on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/18 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Enqueueing ACK on FastEthernet0/0 nbr 192.168.12.1
  Ack seq 18 iidbQ un/rely 0/0 peerQ un/rely 1/2
EIGRP: Sending ACK on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/18 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 1/2
EIGRP: Enqueueing ACK on FastEthernet0/0 nbr 192.168.12.1
  Ack seq 19 iidbQ un/rely 0/0 peerQ un/rely 1/2
EIGRP: Sending ACK on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/19 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 1/2
EIGRP: Received ACK on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/20 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
```

These are the ack packets in response to some of the update packets. If you want to see the whole process we can combine some debugging.

```
John#debug eigrp packets
EIGRP Packets debugging is on
    (UPDATE, REQUEST, QUERY, REPLY, HELLO, IPXSAP, PROBE, ACK, STUB,
SIAQUERY, SIAREPLY)
```

This is how you enable debugging for all EIGRP packets.

```
Jack#clear ip eigrp neighbors
```

Reset the EIGRP neighbor adjacency again…

```
John#
EIGRP: Sending HELLO on FastEthernet0/0
  AS 1, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
EIGRP: Received HELLO on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/0 idbQ 0/0
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 192.168.12.1 (FastEthernet0/0)
is up: new adjacency
John#
EIGRP: Enqueueing UPDATE on FastEthernet0/0 nbr 192.168.12.1 iidbQ un/rely
0/1 peerQ un/rely 0/0
EIGRP: Sending HELLO on FastEthernet0/0
  AS 1, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/1
EIGRP: Requeued unicast on FastEthernet0/0
EIGRP: Received UPDATE on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x1, Seq 25/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Sending UPDATE on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x1, Seq 25/25 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Received ACK on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/25 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Enqueueing UPDATE on FastEthernet0/0 iidbQ un/rely 0/1 serno 1-2
EIGRP: Enqueueing UPDATE on FastEthernet0/0 nbr 192.168.12.1 iidbQ un/rely
0/0 peerQ un/rely 0/0 serno 1-2
EIGRP: Sending UPDATE on FastEthernet0/0
  AS 1, Flags 0x8, Seq 26/0 idbQ 0/0 iidbQ un/rely 0/0 serno 1-2
EIGRP: Enqueueing UPDATE on FastEthernet0/0 nbr 192.168.12.1 iidbQ un/r
John#ely 0/1 peerQ un/rely 0/1
EIGRP: Requeued unicast on FastEthernet0/0
EIGRP: Received ACK on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/26 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/2
EIGRP: FastEthernet0/0 multicast flow blocking cleared
EIGRP: Sending UPDATE on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x8, Seq 27/25 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Received ACK on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/27 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Received UPDATE on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x8, Seq 26/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
EIGRP: Enqueueing ACK on FastEthernet0/0 nbr 192.168.12.1
  Ack seq 26 iidbQ un/rely 0/0 peerQ un/rely 1/0
EIGRP: Sending ACK on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/26 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 1/0
EIGRP: Enqueueing UPDATE on FastEthernet0/0 iidbQ un/rely 0/1 serno 9-9
EIGRP: Enqueueing UPDATE on Fa
EIGRP: Received UPDATE on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x8, Seq 27/27 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Enqueueing ACK on FastEthernet0/0 nbr 192.168.12.1
  Ack seq 27 iidbQ un/rely 0/0 peerQ un/rely 1/1
EIGRP: Sending ACK on FastEthernet0/0 nbr 192.168.12.1
  AS 1, Flags 0x0, Seq 0/27 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 1/1
```

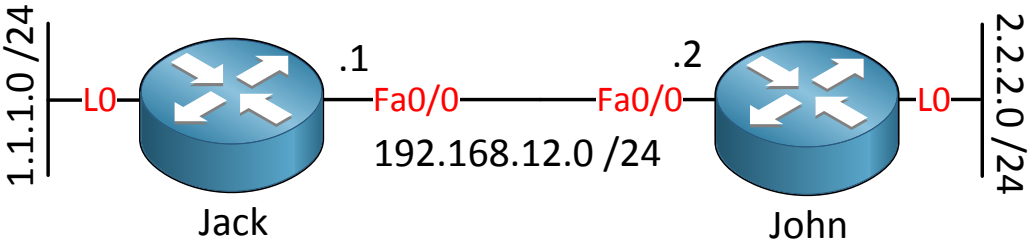
This will show you the entire process of sending hello packets to each other, becoming EIGRP neighbors, exchanging updates and sending acks.

```
John#show ip protocols
Routing Protocol is "eigrp 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 1
  Redistributing: eigrp 1
  EIGRP NSF-aware route hold timer is 240s
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    2.2.2.0/24
    192.168.12.0
  Passive Interface(s):
    Loopback0
    VoIP-Null0
  Routing Information Sources:
    Gateway         Distance      Last Update
    192.168.12.1          90      00:04:42
  Distance: internal 90 external 170
```

This is the last command I want to show you now. **Show ip protocols** is a very useful and powerful command in your CCNP arsenal. It will show you for which networks you are routing, passive interfaces and the administrative distance. See the external administrative distance of 170? We'll talk about that one in the redistribution chapter. This command isn't just for EIGRP it will show you all routing protocols.

Enough debugging for now let's continue by looking at the different EIGRP tables in detail:

- EIGRP Neighbor table
- EIGRP Topology table
- Routing table



Router Jack and John are back and I'm going to show you what the EIGRP neighbor table looks like on a real network:

```
Jack#show ip eigrp neighbors
IP-EIGRP neighbors for process 1
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)       Cnt
Num
0   192.168.12.2            Fa0/0             10 00:06:06   19   200  0  27
```

In the output above I'm looking at the EIGRP neighbor table of router Jack. As you can see we have one neighbor (192.168.12.2) which happens to be router John on interface FastEthernet 0/0. What else do we find here?

- **H (Handle):** Here you will find the order when the neighbor adjacency was established. Your first neighbor will have a value of 0, the second neighbor a value of 1 and so on.
- **Hold Uptime** (sec): this is the **holddown timer** per EIGRP neighbor. Once this timer expires we will drop the neighbor adjacency. The default holddown timer is 15 seconds. On older IOS versions only a hello packet would reset the holddown timer but on newer IOS versions any EIGRP packet after the first hello will reset the holddown timer.
- **SRTT (Smooth round-trip time):** The number of milliseconds it takes to send an EIGRP packet to your neighbor and receive an acknowledgment packet back.
- **RTO (Retransmission timeout):** The amount of time in milliseconds that EIGRP will wait before retransmitting a packet from the retransmission queue to this neighbor.
- **Q Cnt (Q count):** The number of EIGRP packets (Update, Query or Reply) in the queue that are awaiting transmission. Ideally you want this number to be 0 otherwise it might be an indication of congestion on the network.
- **Seq Num (Sequence number):** This will show you the sequence number of the last update,query or reply packet that you received from your EIGRP neighbor.

Excellent so that's how EIGRP stores neighbor information! Our next stop is of course to take a look at the EIGRP Topology table:

```
Jack#show ip eigrp topology
IP-EIGRP Topology Table for AS(1)/ID(1.1.1.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 1.1.1.0/24, 1 successors, FD is 128256
        via Connected, Loopback0
P 2.2.2.0/24, 1 successors, FD is 156160
        via 192.168.12.2 (156160/128256), FastEthernet0/0
P 192.168.12.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/0
```

Now that's a lot of information to look at! Let me break it down for you in chunks:

```
Jack#show ip eigrp topology
```

```
IP-EIGRP Topology Table for AS(1)/ID(1.1.1.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 1.1.1.0/24, 1 successors, FD is 128256
        via Connected, Loopback0
P 2.2.2.0/24, 1 successors, FD is 156160
        via 192.168.12.2 (156160/128256), FastEthernet0/0
P 192.168.12.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/0
```

If you look at the red fonts you can see that we are looking at the EIGRP topology table for **AS (Autonomous System)** number 1. Keep in mind that the AS number has to match on EIGRP routers in order to become neighbors!

```
Jack#show ip eigrp topology
IP-EIGRP Topology Table for AS(1)/ID(1.1.1.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 1.1.1.0/24, 1 successors, FD is 128256
        via Connected, Loopback0
P 2.2.2.0/24, 1 successors, FD is 156160
        via 192.168.12.2 (156160/128256), FastEthernet0/0
P 192.168.12.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/0
```

Look at those codes…Update, Query and Reply should ring a bell since I discussed them a few pages ago. Let's focus on those codes that I didn't explain before:

- **Passive:** Passive is good…we like routing information to be passive which means that we have learned information about a network and there are no changes in the topology table.

- **Active:** Active is not good since it means we have lost information about a certain network and EIGRP doesn't know another way of reaching this network. It will go into active mode and send query packets to ALL its neighbors asking them if they know how to reach this network.

- **Reply Status:** EIGRP will track all the query packets it has sent to neighbors since you need a reply in return. By setting the reply status flag it will do this.

- **Sia Status (Stuck in Active):** This is a bad one…it means that EIGRP has not received a reply to a query packet from one of the neighbors within the allowed time (about 3 minutes). When this happens EIGRP will drop the neighbor adjacency and it will be stuck in active. More on this later!

```
Jack#show ip eigrp topology
IP-EIGRP Topology Table for AS(1)/ID(1.1.1.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 1.1.1.0/24, 1 successors, FD is 128256
        via Connected, Loopback0
P 2.2.2.0/24, 1 successors, FD is 156160
        via 192.168.12.2 (156160/128256), FastEthernet0/0
P 192.168.12.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/0
```

One more thing to show you! Let's look at an entry of a prefix…in this case 2.2.2.0/24 and break it down in pieces:

- **P 2.2.2.0/24**: The **P** stands for **passive** and that's how we like it! As you can see EIGRP stores the network and the subnet mask.
- **1 successor**: The best path to get to a certain network is called the **successor**. It's possible to have backup paths which EIGRP calls the **feasible successor**. In this case there is only one way to get to the destination.
- **FD is 156160**: **FD** stands for **feasible distance** and in plain English we would call this the "total distance" to get to the destination.
- **Via 192.168.12.2:** That's the IP address of the neighbor where we have to send packets to in order to reach the 2.2.2.0/24 network.
- **(156160/128256):** The first value is the feasible distance. The second value is the **advertised distance.** Your EIGRP neighbor will report to you how far it is for him to reach the 2.2.2.0/24 network and this is saved as the advertised distance.
- **FastEthernet0/0**: This one is easy; it's just the interface we are using to send our packets to in order to reach this network.

Does this make sense to you? Understanding the EIGRP topology table is very important for troubleshooting or when we start playing with load balancing.

```
Jack#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-
2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.12.0/24 is directly connected, FastEthernet0/0
     1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
     2.0.0.0/24 is subnetted, 1 subnets
D       2.2.2.0 [90/156160] via 192.168.12.2, 01:31:17, FastEthernet0/0
```

The best information from the EIGRP topology table will be copied to the routing table. What do we find here?

- **D (DUAL or Diffusing Update Algorithm)**: DUAL is the algorithm behind EIGRP which is why we find the D here. Why are we not using the E for EIGRP? That would be nice but the letter E is already in use for EGP (Exterior Gateway Protocol) which is an old routing protocol we don't use anymore.

- **2.2.2.0 [90/156160]:** Do you recognize these values? The first one is the administrative distance which is 90 for EIGRP. 156160 is the feasible distance for this network.

- **Via 192.168.12.2 01:31:17, FastEthernet 0/0**: 192.168.12.2 is the next hop address to reach this network. 01:31:17 is the time how long this network has been in the routing table. In this case that's 1 hour, 31 minutes and 17 seconds. You can also see the interface how to reach this next hop address which is FastEthernet 0/0.

One more thing we have to talk about…the EIGRP metrics! In my introduction I showed you really low values since it's easier to explain EIGRP that way. In the examples above you could see that the feasible and advertise distance values are a bit higher which makes them annoying to work with. Let's take a good look at the EIGRP metrics and the formula.

Let's start with the formula that EIGRP uses:

EIGRP Metric = 256((K1*Bw) + (K2*Bw)/(256-Load) + K3*Delay)*(K5/(Reliability + K4)))*

Ewww…that looks bad! Don't worry…you don't have to remember this formula! If you look at it you can see that it incorporates bandwidth, load, delay and reliability and you can see K1, K2, K3, K4 and K5 values.

If you studied CCNA you might have seen and/or learned the following list:

- **Bandwidth (K1)**
- **Load (K2)**
- **Delay (K3)**
- **Reliability (K4)**
- **MTU (K5)**

Technically this is incorrect. K1 doesn't correspond 1:1 with bandwidth, K2 doesn't correspond 1:1 with load, delay not with K3 etc. These K values are only numbers to **scale numbers** in the metric calculation.

We can see what K values are enabled or disabled by default:

```
Jack#show ip protocols
Routing Protocol is "eigrp 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 1
  Redistributing: eigrp 1
  EIGRP NSF-aware route hold timer is 240s
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    1.1.1.0/24
    192.168.12.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    192.168.12.2          90      00:14:30
  Distance: internal 90 external 170
```

In this example where I used the command **show ip protocols** you can see which K-values are enabled by default. Only K1 and K3 are enabled by default.

Let's walk through the different metric components to see what they are:

Bandwidth:

```
Jack#show interfaces fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is cc02.58a9.0000 (bia cc02.58a9.0000)
  Internet address is 192.168.12.1/24
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
```

If you use the **show interface FastEthernet 0/0** command you can see the interface information. The example above only shows part of the output. You can see the bandwidth is 100000 Kbit which is a 100Mbit interface.

```
Jack(config)#interface fa0/0
Jack(config-if)#band
Jack(config-if)#bandwidth ?
  <1-10000000>  Bandwidth in kilobits
  inherit       Specify that bandwidth is inherited
  receive       Specify receive-side bandwidth

Jack(config-if)#bandwidth 500
```

Bandwidth is a *static* value which can be changed by using the **bandwidth** command. Keep in mind this doesn't change the *actual* bandwidth of the interface! This command is ONLY used to influence routing protocols like EIGRP. It's not like you can slow down electric signals through a wire…if you want to limit the traffic on an interface you'll need QoS (Quality of Service) which is a story for another day!

```
Jack#show interfaces fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is cc02.58a9.0000 (bia cc02.58a9.0000)
  Internet address is 192.168.12.1/24
  MTU 1500 bytes, BW 500 Kbit, DLY 100 usec,
```

Here you see the result of changing the bandwidth on the interface. Something to remember is that EIGRP will **use the lowest bandwidth** in the path from A to B (since this is the bottleneck).

Load:

```
Jack#show interfaces fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is cc02.58a9.0000 (bia cc02.58a9.0000)
  Internet address is 192.168.12.1/24
  MTU 1500 bytes, BW 500 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
```

Load will show you how busy the interface is based on the packet rate and the bandwidth on the interface. This is a value that can change over time so it's a *dynamic* value.

Delay:

```
Jack#show interfaces fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is cc02.58a9.0000 (bia cc02.58a9.0000)
  Internet address is 192.168.12.1/24
  MTU 1500 bytes, BW 500 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
```

Delay reflects the time it will take for packets to cross the link and is a *static* value. Cisco IOS will have default delay values for the different types of interface. A FastEthernet interface has a default delay of 100 usec.

```
ack(config)#interface fa0/0
Jack(config-if)#delay ?
  <1-16777215>  Throughput delay (tens of microseconds)

Jack(config-if)#delay 50
```

If you use the **delay** command you can change this value to influence routing protocols like EIGRP. It doesn't actually change the delay for this interface but it is only used to influence routing protocols.

```
Jack#show interfaces fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is cc02.58a9.0000 (bia cc02.58a9.0000)
  Internet address is 192.168.12.1/24
  MTU 1500 bytes, BW 500 Kbit, DLY 500 usec,
```

Above you see the delay that I changed. EIGRP will **accumulate all the delay values** in the path from A to B.

Reliability:

```
Jack#show interfaces fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is cc02.58a9.0000 (bia cc02.58a9.0000)
  Internet address is 192.168.12.1/24
  MTU 1500 bytes, BW 500 Kbit, DLY 500 usec,
     reliability 255/255, txload 1/255, rxload 1/255
```

Reliability at 255/255 is 100%. This means that you don't have issues on the physical or data-link layer. If you are having issues this value will decrease. Since this is something that can change it's a *dynamic value.*

MTU:

```
Jack#show interfaces fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is cc02.58a9.0000 (bia cc02.58a9.0000)
  Internet address is 192.168.12.1/24
  MTU 1500 bytes, BW 500 Kbit, DLY 500 usec,
     reliability 255/255, txload 1/255, rxload 1/255
```

MTU or Maximum Transmission Unit is being exchanged between EIGRP neighbors but not used for the metric calculation.

By default only K1 and K3 are enabled and we don't use K2 or K4. This means that only bandwidth and delay are used in the formula.

Why not? Because loading and reliability are *dynamic* values and they can change over time. You don't want your EIGRP routers calculating 24/7 and sending updates to each other just because the load or reliability of an interface has changed. We want routing protocols to be nice and quiet and only base their routing decisions on *static* values like bandwidth and delay. If you only use those two static values our EIGRP routers don't have to do any recalculation unless an interface goes down or a router died.

Since only K1 and K3 are enabled we can simplify the EIGRP formula:

Metric = bandwidth (slowest link) + delay (sum of delays)

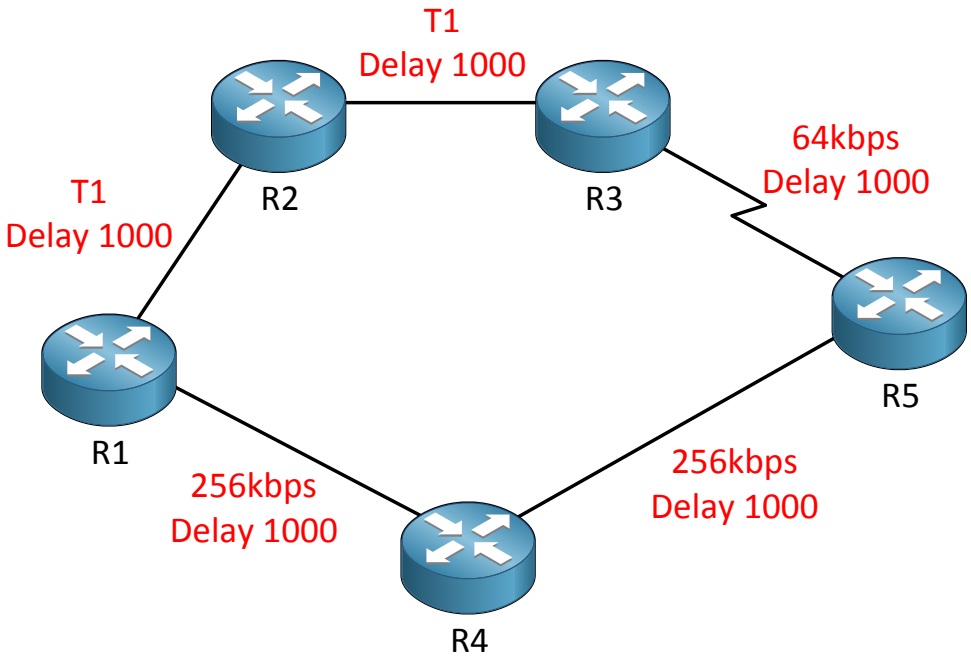
- Bandwidth: [10^7 / minimum bandwidth in the path] * 256.
- Delay: sums of delays in the path multiplied by 256 (in tens of microseconds).

So the formula looks like:

Metric = (10^7 / minimum bandwidth) * 256 + (sum of delays) * 256

The multiplication of 256 is done so EIGRP is compatible with IGRP (the predecessor of EIGRP).

Let me show you an example so we can break down this formula:



We are looking at R1 and calculating the distance to get to R5. As you can see there is an upper path with some T1 interfaces and a 64kbps link. The path below has two 256kbps links.

A T1 interface has a bandwidth of 1.554Mbit which is obviously better than 256kbps but the bottleneck in the upper path is our 64kbps link. Let's throw these numbers for the upper path in the EIGRP metric formula and see what happens:

The lowest bandwidth in the upper path is our 64kbps link so the EIGRP bandwidth calculation will look like this:

Bandwidth = (10^7 / slowest link) * 256
Bandwidth = (10,000,000 / 64) * 256 = 156,250 * 256 = 40,000,000

Now let's look at the delay calculation for the upper path:

Delay = [sum of delays] * 256
Delay = [1000+1000+1000] * 256
Delay = 768,000

Let's add those numbers together and we'll have the total metric:

Metric = bandwidth + delay
Metric = 40,000,000 + 768,000
Metric = 40,768,000

Having fun yet? Let's do the lower path as well!

The lowest bandwidth in the lower path is 256kbps link so the EIGRP bandwidth calculation will look like this:

Bandwidth = (10^7 / slowest link) * 256
Bandwidth = (10,000,000 / 256) * 256 = 39062.5 * 256 = 10,000,000

Now let's look at the delay calculation for the lower path:

Delay = [sum of delays] * 256
Delay = [1000+1000] * 256
Delay = 512,000

Let's add those numbers together and we'll have the total metric:

Metric = bandwidth + delay
Metric = 10,000,000 + 512,000
Metric = 10,512,000

Upper path metric = 40,768,000
Lower path metric = 10,512,000

The lower metric will be installed as the successor route in the routing table so the lower path will be used in this example.

Maybe you are wondering what the formula looks like if you would enable loading (K2) and reliability (K4) as well, well here it is:

Metric = [K1*bandwidth + ((K2*bandwidth)/(256-load))+K3*delay]

If MTU (K5) is not equal to 0:

Metric = Metric*[K5/(reliability+K4)]

Phew! Does this make your head spin? I think you and I both agree we should let EIGRP do the metric calculations and not us (that's why we have routers right!). The important lesson I wanted to show you here is that EIGRP uses the **slowest bandwidth** in the path and the **sum of delays**. You don't have to know this formula by heart but **understand** it. No need to do any manual calculations on the exam!

The metrics in EIGRP are a pain to work with since the values are so LARGE! If you want to practice with EIGRP you can try to disable all the K-values except K3. This will make EIGRP only use delay as metric. The metric values will be much lower and easier to work with since you don't have to think about the lowest bandwidth in the path. I like to do this when I'm teaching people how to calculate feasible successors and configure EIGRP load balancing.

Feel like playing with the metrics and some load balancing? Try the following labs:

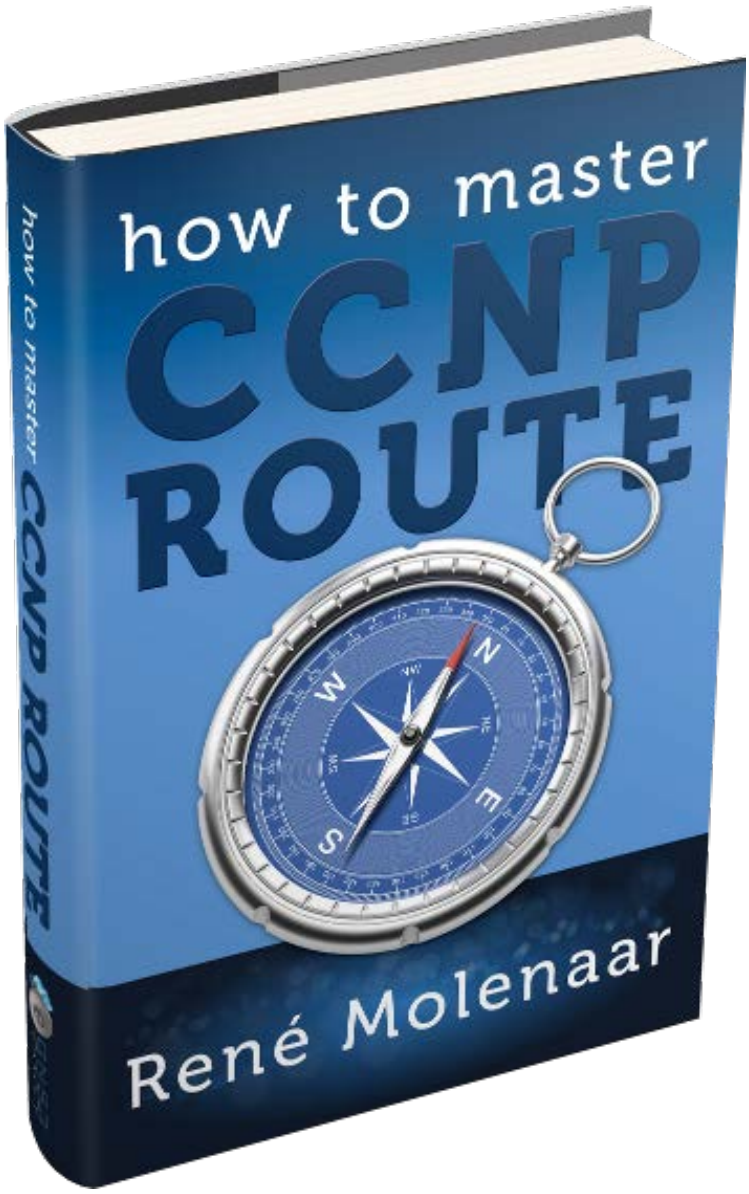
http://gns3vault.com/EIGRP/eigrp-maximum-path-and-variance.html

http://gns3vault.com/EIGRP/eigrp-intermediate.html

Do you enjoy reading this sample of How to Master CCNP ROUTE ?

Click on the link below to get the full version.

Get How to Master CCNP ROUTE Today



3. EIGRP Summarization

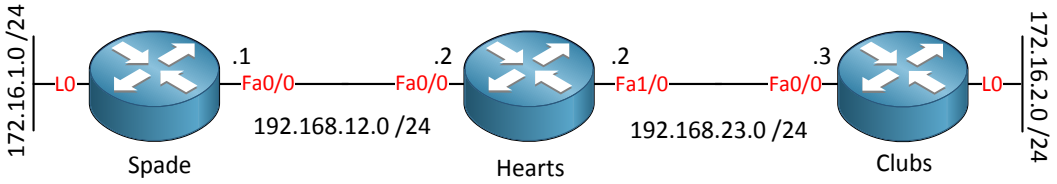
If you studied and passed CCNA you probably have an idea why we use summaries:

- Decrease the size of the routing table.
- Fewer routing updates.

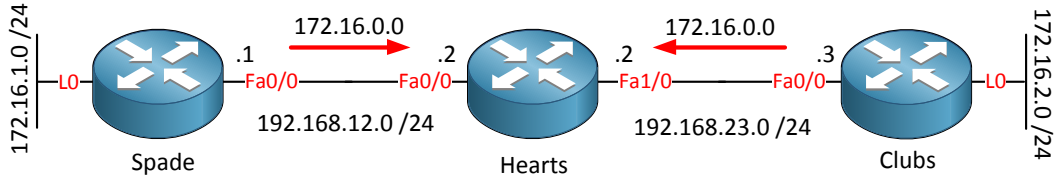
EIGRP has two ways of summarizing networks:

- Automatic summarization:
  - Subnets are summarized to the classful network.
  - This is the default for EIGRP.
- Manual summarization.

You should disable EIGRP automatic summarization since it can cause issues with routing.



Look at the topology above. We have 3 routers and we are configuring EIGRP. Note the 172.16.1.0 and 172.16.2.0 networks. EIGRP will summarize to the classful network by default.

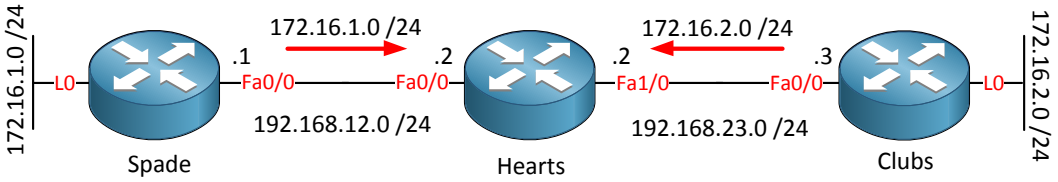


Router Spade and router Clubs don't send the subnet mask along with the routing update so it will advertise the classful network which is 172.16.0.0 in this case. So what happens with router Hearts? It thinks it can reach the 172.16.0.0 network by sending packets either left or right and if the metric is equal it will try to load-balance. Obviously this is going to cause problems.

```
Spade(config)#router eigrp 1
Spade(config-router)#no auto-summary
```

```
Clubs(config)#router eigrp 1
Clubs(config-router)#no auto-summary
```

Type in the **no auto-summary** command to make sure EIGRP behaves classless and sends the subnet mask along.

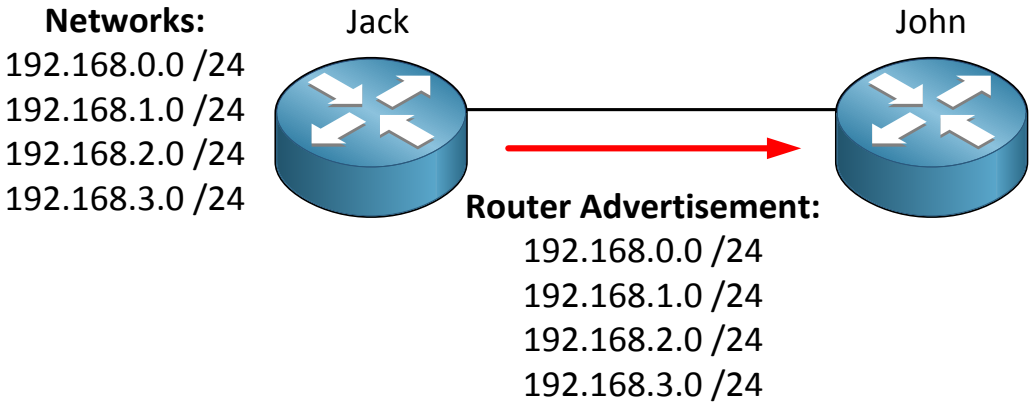


This is what we want to achieve. Router Spade and Clubs will send the subnet mask with their EIGRP update packets.

Let's take a look at manual summarization which is more fun. In the picture below we have router Jack and John, router Jack has the following networks configured:

192.168.0.0 / 24
192.168.1.0 / 24
192.168.2.0 / 24
192.168.3.0 / 24

When we configure EIGRP, all 4 networks will be advertised and seen in the routing table of Router John:



Router John's routing table:

```
John#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-
2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.12.0/24 is directly connected, FastEthernet0/0
D    192.168.0.0/24 [90/156160] via 192.168.12.1, 00:00:06, FastEthernet0/0
D    192.168.1.0/24 [90/156160] via 192.168.12.1, 00:00:06, FastEthernet0/0
D    192.168.2.0/24 [90/156160] via 192.168.12.1, 00:00:06, FastEthernet0/0
D    192.168.3.0/24 [90/156160] via 192.168.12.1, 00:00:06, FastEthernet0/0
```

Let's configure a summary on router Jack to decrease the size of router John's routing table. Creating summaries for EIGRP is easy since you can do it on any interface you like.

```
Jack(config)#interface fastEthernet 0/0
Jack(config-if)#ip summary-address eigrp 1 192.168.0.0 255.255.252.0
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 192.168.12.2 (FastEthernet0/0)
is resync: summary configured
```

I summarized those 4 networks into 192.168.0.0 /22. You need to specify the AS number and the subnet mask to send along the network. As you can see it's resyncing with router John.

```
John#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-
2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.12.0/24 is directly connected, FastEthernet0/0
D    192.168.0.0/22 [90/156160] via 192.168.12.1, 00:00:39, FastEthernet0/0
```

This is what router John looks like now. We reduced its routing table from 4 entries to just 1 entry.

```
Jack#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-
2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.12.0/24 is directly connected, FastEthernet0/0
C    192.168.0.0/24 is directly connected, Loopback0
C    192.168.1.0/24 is directly connected, Loopback1
C    192.168.2.0/24 is directly connected, Loopback2
C    192.168.3.0/24 is directly connected, Loopback3
D    192.168.0.0/22 is a summary, 00:01:09, Null0
```

Router John is not the only one who had a routing table metamorphose. Look at router Jack above and check out the last entry. 192.168.0.0/22 has been created sending packets to Null0. Our Null0 interface is like a black hole sucking up packets never to return again…ouch! Why does EIGRP do this and what is it good for? Let me show you another example.

```
Jack(config)#interface fastEthernet 0/0
Jack(config-if)#ip summary-address eigrp 1 192.168.0.0 255.255.0.0
```

The previous example was an A+ example of summarization because I summarized the following networks:

192.168.0.0 /24
192.168.1.0 /24
192.168.2.0 /24
192.168.3.0 /24

Into 192.168.0.0 /22 which is a perfect match.

If you look at the example above I made a change. I created the summary 192.168.0.0 /16 which router Jack is advertising towards router John. This is a C- example! The summary is working but I'm advertising a huge range of networks that I don't have.

Let's look at router John's routing table:

```
John#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-
2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.12.0/24 is directly connected, FastEthernet0/0
D    192.168.0.0/22 [90/156160] via 192.168.12.1, 00:02:29, FastEthernet0/0
D    192.168.0.0/16 [90/156160] via 192.168.12.1, 00:00:34, FastEthernet0/0
```

There are 2 things to mention here:

- We now have a 192.168.0.0 /16 entry in our routing table.
- You can see that my latest summary 192.168.0.0 /16 does not overwrite the old summary. You have to remove the old one yourself.

So what happens when we send a ping towards an IP address within the 192.168.0.0 /16 address space but not configured on any interface?

```
John#ping 192.168.200.20

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.200.20, timeout is 2 seconds:
U.U.U
Success rate is 0 percent (0/5)
```

As you can see router Jack is telling us through ICMP that this IP address is unreachable.

```
Jack(config)#access-list 100 permit ip any 192.168.200.20 0.0.0.0
Jack(config)#exit
Jack#debug ip packet 100
IP packet debugging is on for access list 100
```

Let's switch over to router Jack and do a debug. **Debug ip packet** is VERY useful but you need to use an access-list otherwise you drown in information.

```
John#ping 192.168.200.20

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.200.20, timeout is 2 seconds:
U.U.U
Success rate is 0 percent (0/5)
```

Let's send those pings again…

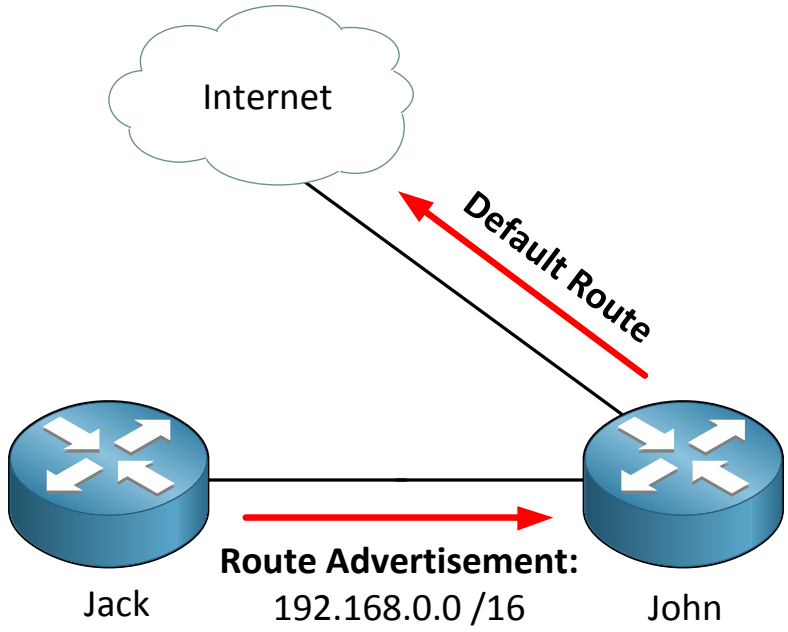
```
Jack#
```

```
IP: tableid=0, s=192.168.12.2 (FastEthernet0/0), d=192.168.200.20 (Null0),
routed via RIB
Jack#
IP: tableid=0, s=192.168.12.2 (FastEthernet0/0), d=192.168.200.20 (Null0),
routed via RIB
Jack#
IP: tableid=0, s=192.168.12.2 (FastEthernet0/0), d=192.168.200.20 (Null0),
routed via RIB
```

Those packets are boldly going where no packet has gone before…only this time there are no strange new worlds to discover….those packets are gone forever!

Why does EIGRP work like this? There's a very good reason. What if there was a 3rd router in our topology and router Jack has a default route pointing to this router?

We will end up forwarding packets for 192.168.200.20 to the default route with the chance of creating a routing loop.



Let's spice up our topology to see why we need this Null0 interface. Router John has another interface connected to the Internet. Router John has a default Route to the Internet.

Since we want Internet access on router Jack as well we configure a default route on router Jack towards router John.

```
Jack(config)#ip route 0.0.0.0 0.0.0.0 192.168.12.2
```

This is what the default route on router Jack pointing to John looks like.

```
Jack#show ip route | exclude C
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-
2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 192.168.12.2 to network 0.0.0.0

S*   0.0.0.0/0 [1/0] via 192.168.12.2
D    192.168.0.0/22 is a summary, 00:09:40, Null0
D    192.168.0.0/16 is a summary, 00:07:45, Null0
```

This is what will happen if we don't have the Null0 interface:

1. Router John has the 192.168.0.0 /22 summary in its routing table.
2. Router John sends an IP packet to 192.168.200.20 and forwards it to router Jack.
3. Router Jack doesn't have the 192.168.200.X network in its routing table but does have a default route.
4. Router Jack will forward the IP packet to router John.
5. Uh-oh…we have a routing loop!

IP Packets have a TTL (Time to Live) field so they won't bounce around forever but it's not a good thing.

Thanks to our Null0 interface this is not going to happen, watch this:

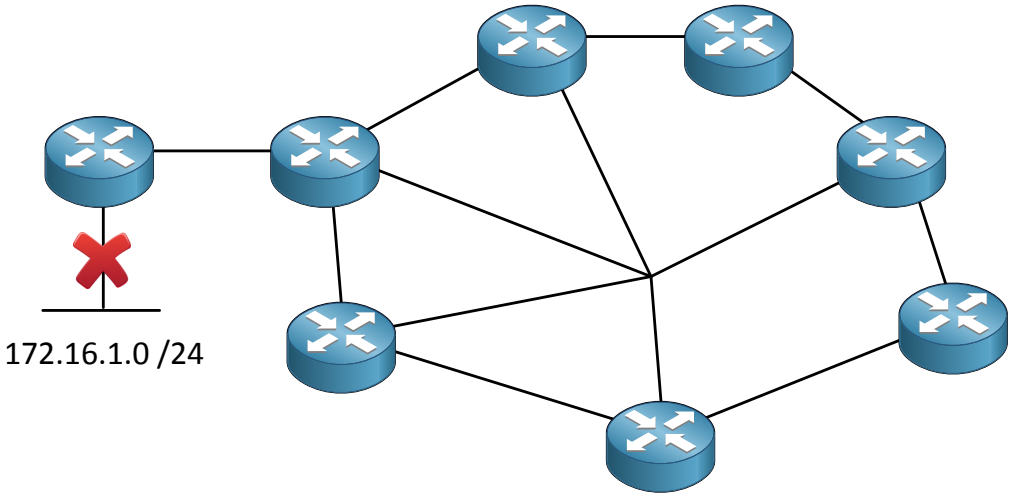
```
John#ping 192.168.200.20

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.200.20, timeout is 2 seconds:
U.U.U
Success rate is 0 percent (0/5)
```

We are sending another ping from router John to router Jack.

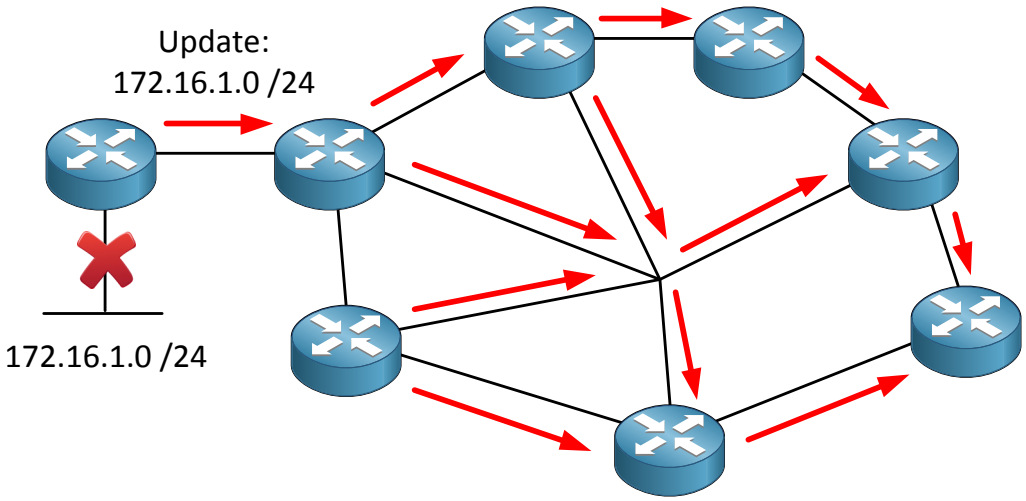
```
Jack#show ip route | exclude C
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-
2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 192.168.12.2 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 192.168.12.2
D     192.168.0.0/22 is a summary, 00:11:19, Null0
D     192.168.0.0/16 is a summary, 00:09:25, Null0
```

Router Jack will do a lookup in its routing table to see if anything matches 192.168.200.20. Our entry with 192.168.0.0/22 is more specific than 0.0.0.0/0 so that's the one we are going to use. Packets will be forwarded to Null0 and are gone! No more routing loops…

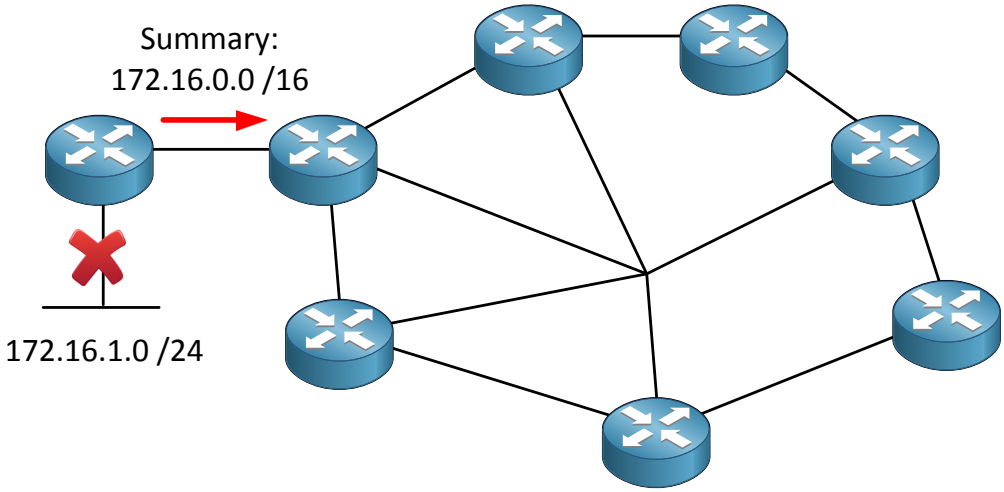
Creating summaries has one more advantage besides reducing the size of routing tables. You will also have less routing updates on your network.



Look at the topology above. We are running EIGRP on all routers. The router on the left side has an interface that's going down.



Our router will send an update to its EIGRP neighbor which will pass it along to the other routers running EIGRP. The whole network is updating itself because just a single interface went down. Summaries can help us here.



If we had a summary configured on the left router to its EIGRP neighbor nothing would happen as soon as our interface goes down. All the routers on the right side have 172.16.0.0 /16 in their routing table and that's it.

What else can I tell you about EIGRP summarization?

- As soon as you delete the last specific route of the summary, your summary will be deleted from the routing table.
- The lowest metric you have for a specific route will be used for the summary route.
- If you want you can specify a different AD (administrative distance) for your summary.

That's all I have for you on EIGRP summarization! Did you enjoy this? We still have a couple of EIGRP chapters left!

If you want to see EIGRP summarization in action you should look at the following labs:

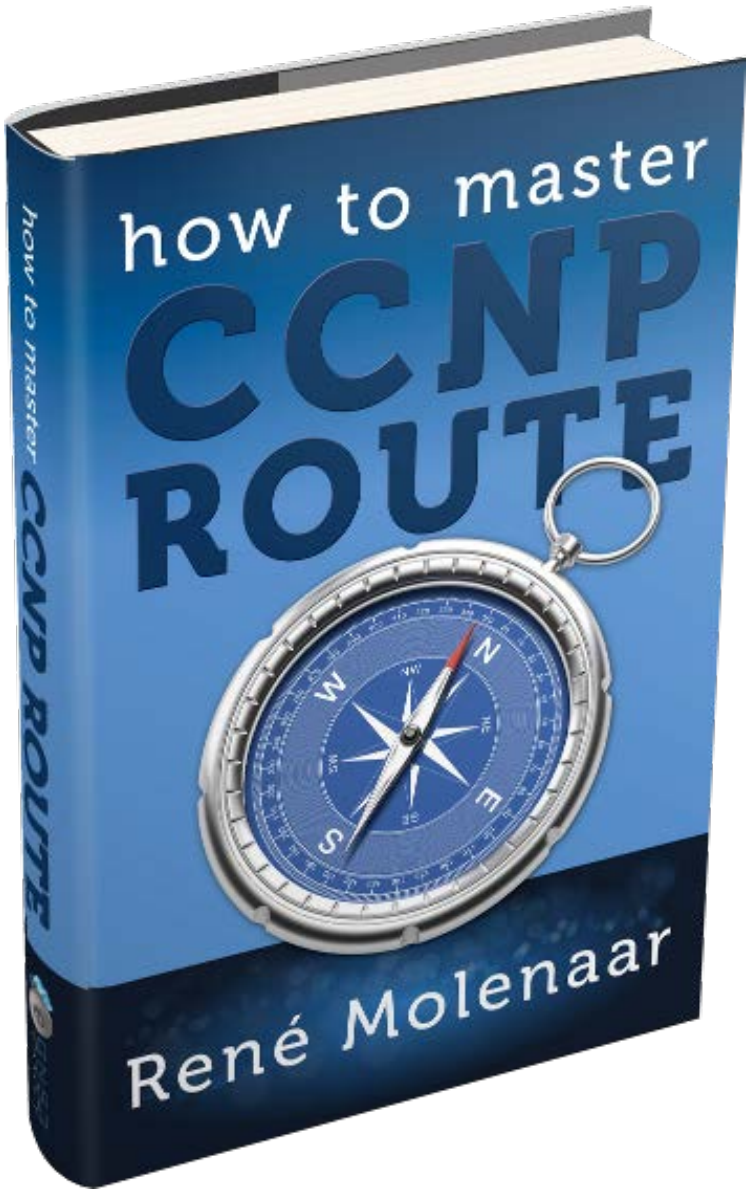
http://gns3vault.com/EIGRP/eigrp-auto-summarization.html

http://gns3vault.com/EIGRP/eigrp-summarization.html

Do you enjoy reading this sample of How to Master CCNP ROUTE ?

Click on the link below to get the full version.

Get How to Master CCNP ROUTE Today

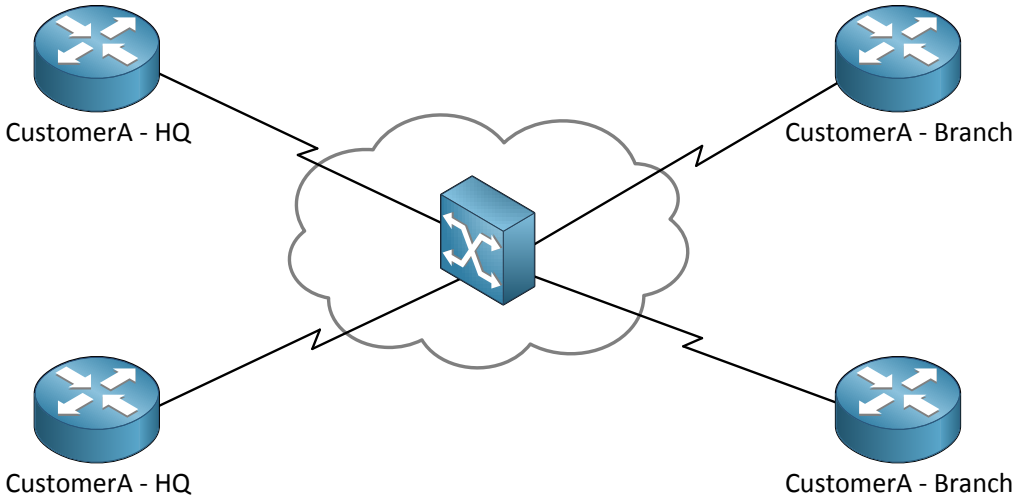


4. EIGRP over Frame-Relay

If you studied CCNA you have seen frame-relay and the following concepts should ring a bell for you:

- NBMA (Non broadcast multi-access network)
- Inverse ARP
- Point-to-point and point-to-multipoint
- Split-horizon issues

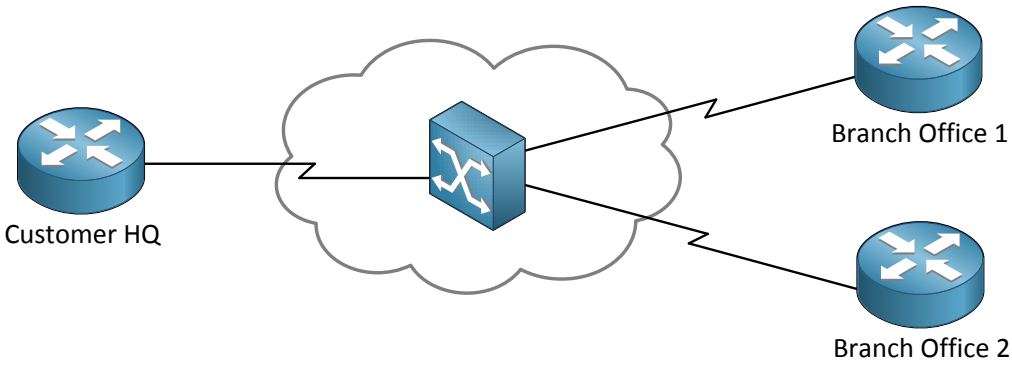
Frame-relay is one of those topics that you might not encounter in real life very often but is still heavily tested on Cisco exams. For those of you who are new to frame-relay or a little fuzzy on the subject I'm going to start with an overview.



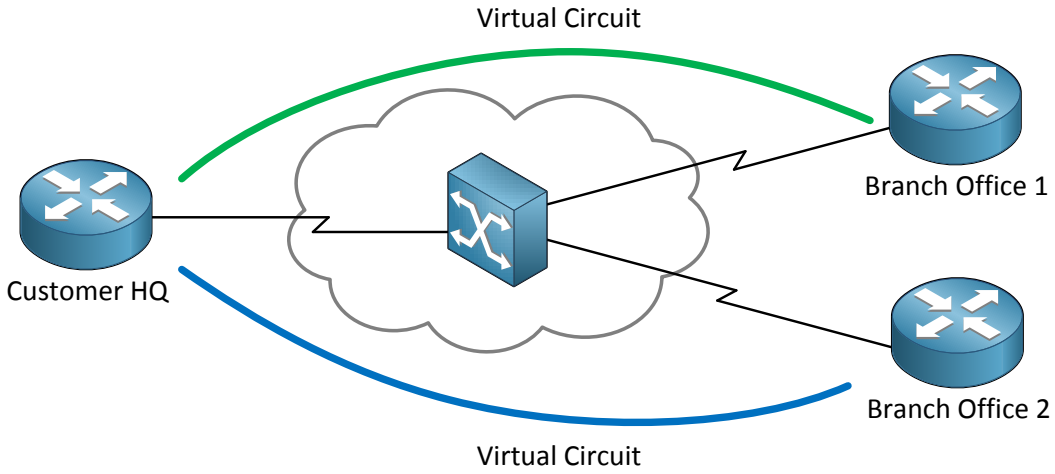
This is a picture of frame-relay. The idea behind frame-relay is that you have a single infrastructure from the service provider and multiple customers are connected to it, effectively sharing everything.

In the middle you see a cloud with an icon. This icon is the frame-relay switch. The cloud is called the **frame-relay cloud** and the reason it has this name is because for us as customer it's unknown what happens in the frame-relay cloud. This is the service provider's infrastructure and we really don't care what happens there…we are the customer and all we want is connectivity!

What else do you see? There are two customers (A and B) and each of them has a HQ (Headquarters) and a branch office.



One more picture, here's a frame-relay network with three routers from one company. There's a router at the headquarters and we have two branch offices. All of them are connected to the frame-relay cloud.



We call our service provider since we want connectivity and the first question they'll ask us is which sites should be connected? In the example above you can see two **virtual circuits**, the red and blue one. With frame-relay there's a difference between the physical and logical connections. The physical connection is just the serial cable which is connected to the provider. Our logical links are virtual circuits. As you can see there is a virtual circuit from branch Office 1 to the HQ router and another one from branch office 2 to the HQ routers.

This means that we can send traffic through our virtual circuits between:

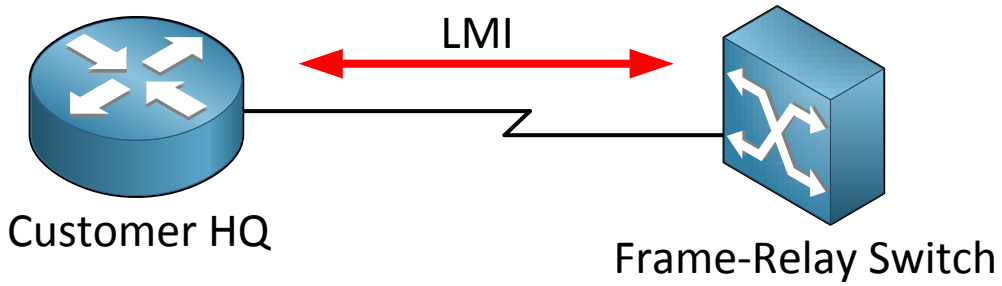
- Branch office 1 and HQ.
- Branch office 2 and HQ.

There is no virtual circuit between branch office 1 and branch office 2. Does this mean there is no connectivity between them? No you can still have connectivity between them by sending data to the HQ router! Of course you can get another virtual circuit between branch office 1 and branch office 2 but you'll have to pay for it. Virtual circuits are also called **PVC (Permanent Virtual Circuit)**.

You also pay for a certain speed called the **CIR (Committed Information Rate).** The cool thing about frame-relay is that when no other customers are using the frame-relay network

it's possible you get a higher speed than what you paid for…the CIR however is a speed that is guaranteed.

How do we know if a PVC is working or not?



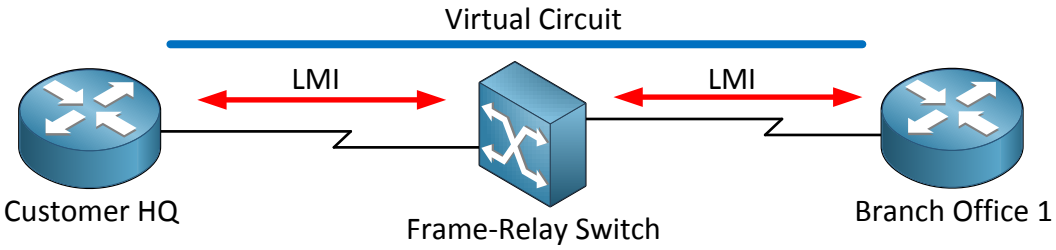
Frame-relay uses something called **LMI** which stands for **Local Management Interface**. LMI has two functions:

- It's a **keepalive** mechanism.
- It tells us if the PVC is active or inactive.
- It also gives us a **DLCI** (Data Link Connection Identifier). I'll get back to this in a bit

There are 3 types of LMI. They all do the same thing but there are three standards which are not compatible with each other. Whatever you choose make sure it's the same between two devices:

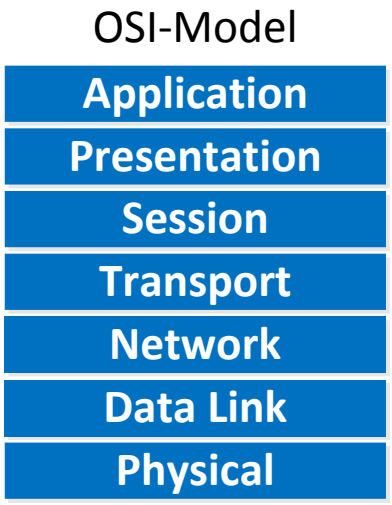
- ✓ Cisco
- ✓ ANSI T1.617 Annex D
- ✓ ITU-T Q.933 Annex A

So if you pick Cisco on one side, use Cisco on the other side as well.

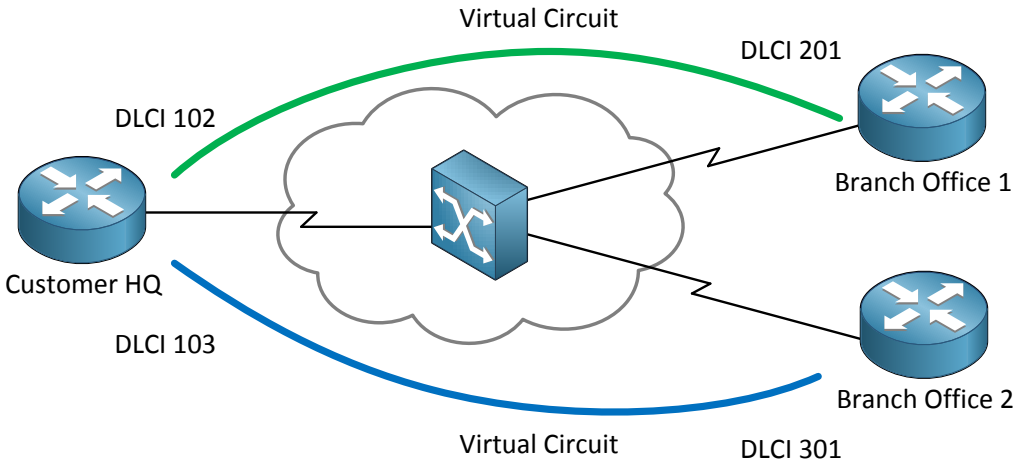


Here's an example of LMI in action. In the middle we have the frame-relay switch. LMI packets are sent between Router A and the frame-relay switch and router B and the frame-relay switch. The frame-relay switch tells our routers that the PVC is active.

What else do you need to know about frame-relay? Let me throw the OSI-model at you:



WAN protocols describe the physical (layer 1) and data link (layer 2) layer. What does frame-relay use on the data link layer? We don't use MAC addresses since that's Ethernet but we do have something else called a **DLCI** (**Data Link Connection Identifier).**



For each PVC you will get a DLCI per router. In our example above you can see that for the PVC between router HQ and branch office 1 we have DLCI 102 on the HQ router and DLCI 201 on the branch office 1 router.

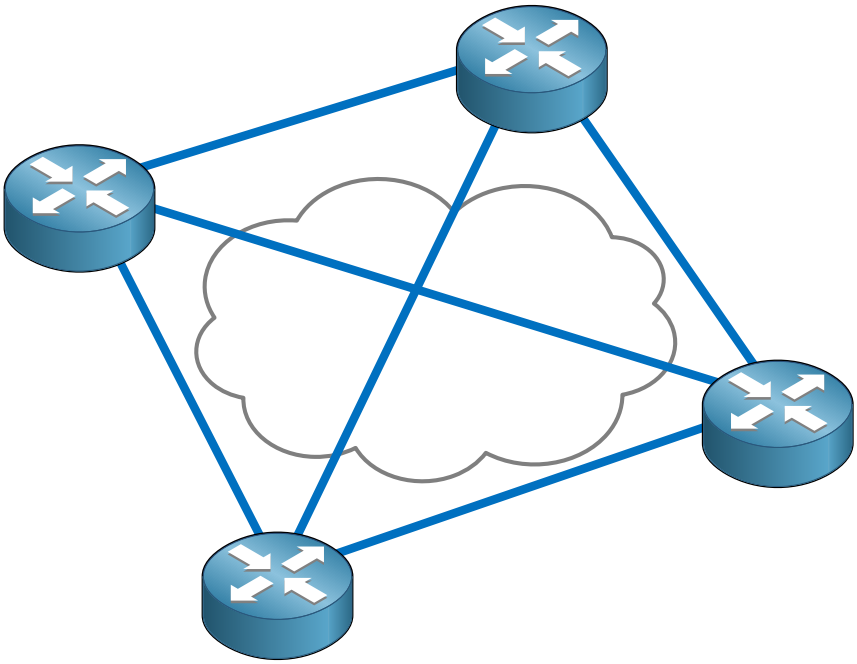
Between router HQ and router branch office 2 we have DLCI 103 on HQ and DLCI 301 on branch office 2. Our DLCI is nothing more but a unique identifier for the data link layer per PVC.

Now there is an important concept to grasp and remember about DLCI. DLCI's are only **locally known** to the router! Your router does **not know** the DLCI of the router on the other side. This is different if you compare it to Ethernet. In our Ethernet world you need to know the MAC address of the computer on the other side in order to send something to it.

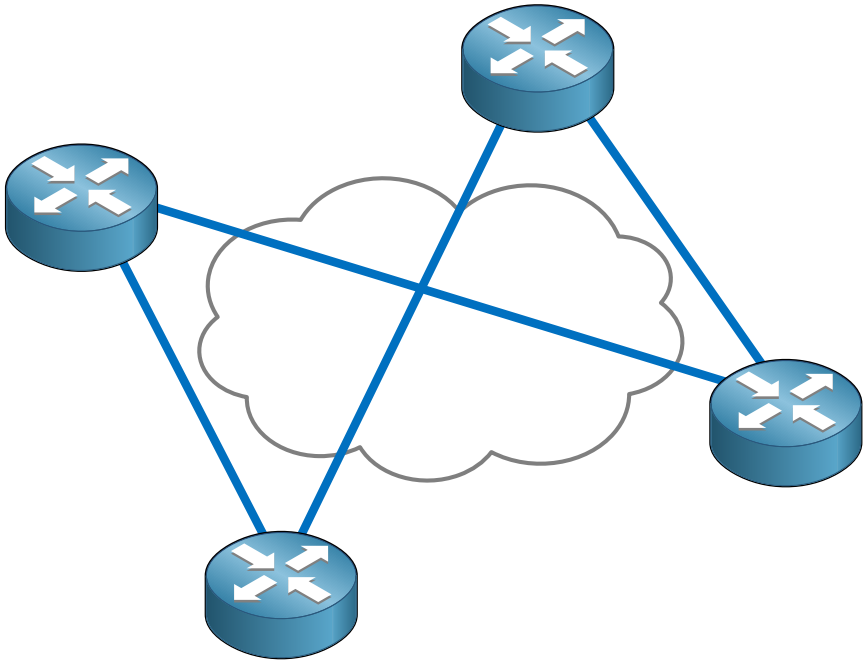
This is just like taking a train. If you are at the train station you walk to the correct train platform and take the train. You have no idea on which train platform you will arrive and you don't care.

Frame-relay supports multiple topologies:

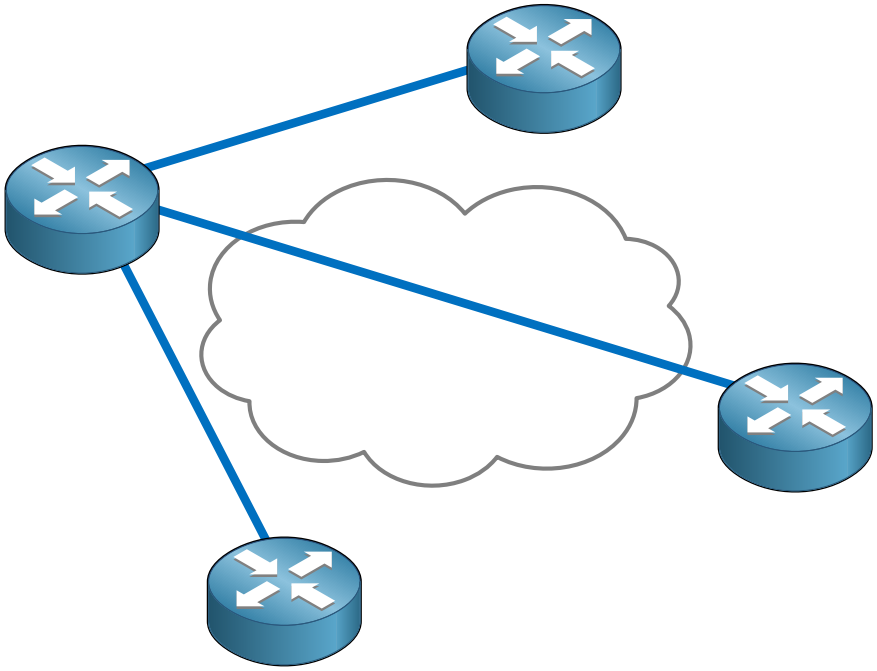
- **Full-mesh**
- **Partial-mesh**
- **Hub and Spoke**



This is our full-mesh topology. As you can see there is a PVC between every router.



This is partial-mesh. The more important routers will have multiple connections to others.



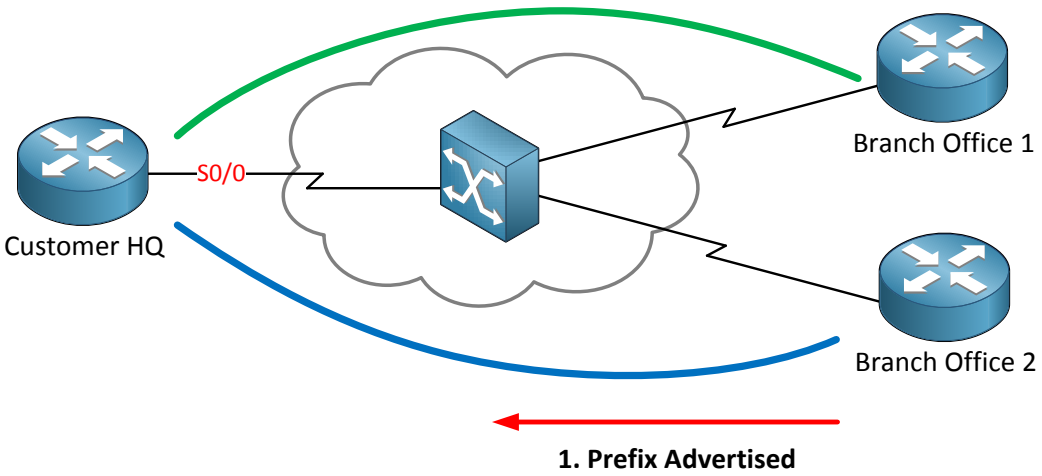
This is the hub and spoke model. The router on the left is our hub and the other routers are spokes. If the spokes want to communicate with each other they'll have to send traffic

towards the hub router.

Frame-relay is **NBMA (non-broadcast multi-access)**

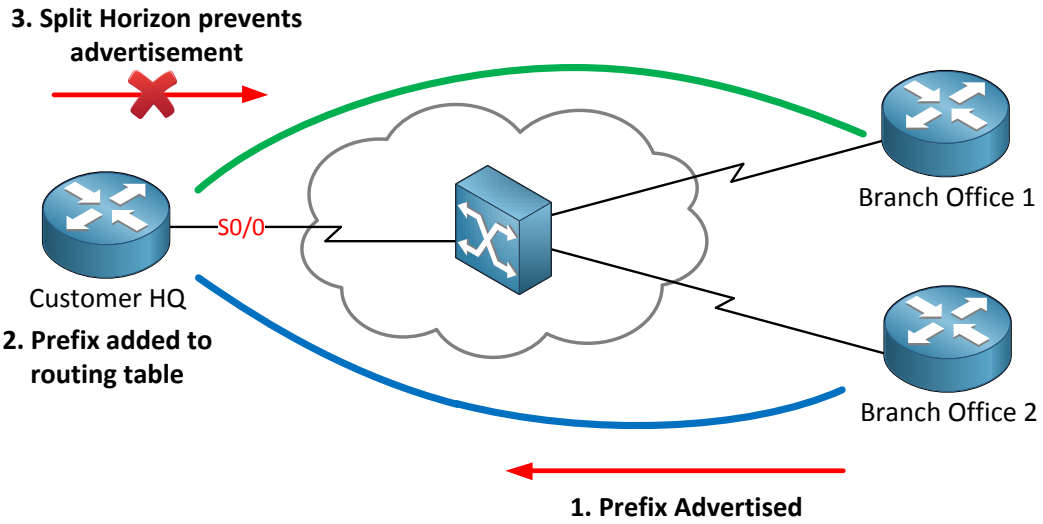
Keep this in mind. What it means is that frame-relay is multi-access since all routers can access the network but you are unable to send broadcasts over the frame-relay network. No broadcast also means you are unable to send multicast traffic. No multicast means you'll be in trouble with routing protocols. Rip version 2, OSPF and EIGRP all use multicast. Does this mean you can't use routing protocols with frame-relay? Well no but it's a bit tricky:

- RIP, OSPF and EIGRP can also use unicast instead of multicast.
- There is a method to "emulate" broadcasts over your frame-relay network.



What other problems might we encounter with frame-relay and routing? Do you remember the characteristics of distance vector routing protocols (RIP and EIGRP)? Split-horizon anyone?

In the picture above I have configured EIGRP on all the routers. Router branch office 1 is sending routing information towards router Customer HQ. If we look at the routing table we see this routing information on router HQ.

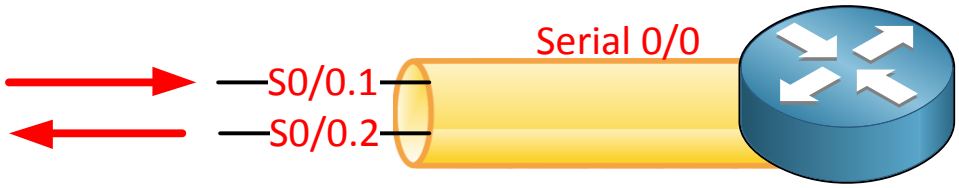


Do you remember the split-horizon rule? Whatever you learn from your neighbor you don't advertise back to them. To be more specific: **whatever you learn on an interface you don't advertise it back out on the same interface.**

We are using two PVC's but on router HQ there is still only one physical interface. Split-horizon will prevent the advertisement of routing information towards router branch office 2.

How can we solve this problem?

- You can disable split-horizon (the default on physical interfaces).
- You can use sub-interfaces.



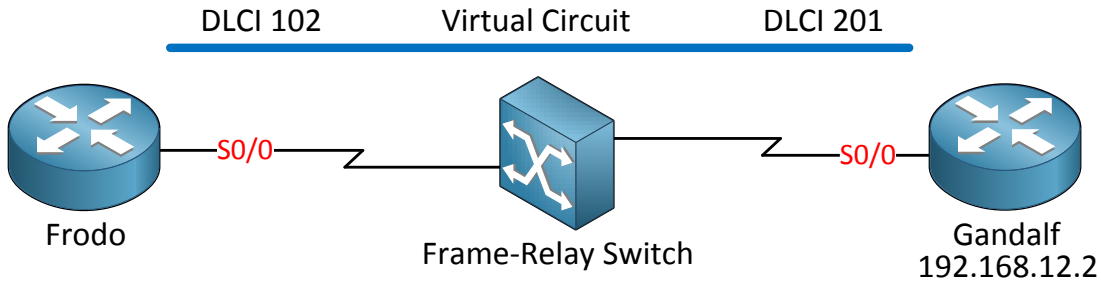
If you use a sub-interface you don't have the split-horizon problem since you are learning routing information on serial0/0.1 and advertising it out of serial0/0.2

Frame-relay can use **point-to-point** sub-interfaces or **point-to-multipoint** sub-interfaces. If you use point-to-point it will solve your split-horizon problem but you'll need to use a different IP subnet per PVC. Point-to-multipoint means you have the split-horizon problem but you can use a single IP subnet for all PVCs.

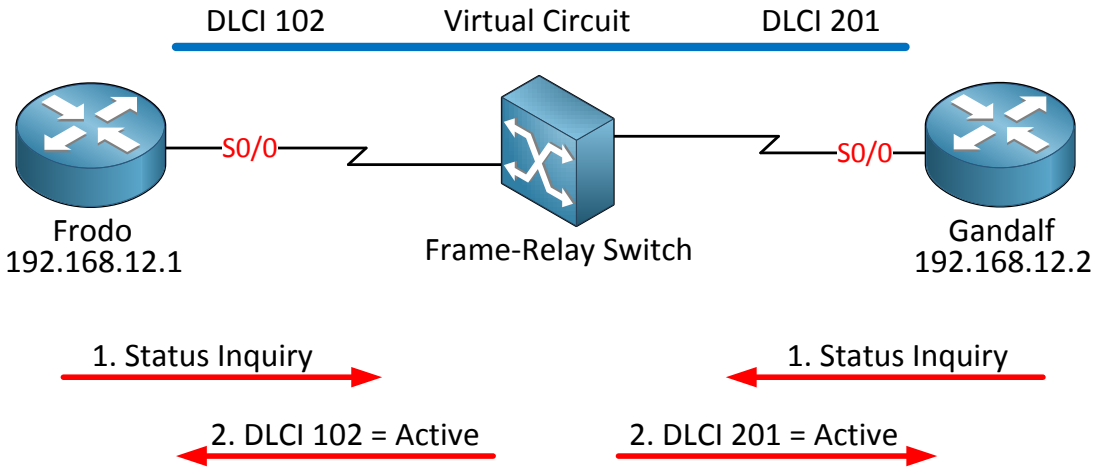
Remember ARP (address resolution protocol)? When we use ARP for Ethernet we need to learn the MAC address of the computer we want to send something to. ARP effectively maps the destination IP address to the destination MAC address.

Frame-relay uses **inverse ARP** and is a bit different. Remember my story about the train platform and how your router only knows it's local DLCI? You don't know the DLCI of the other side. Inverse ARP is going to map your **local DLCI to the IP address of the other**

side:

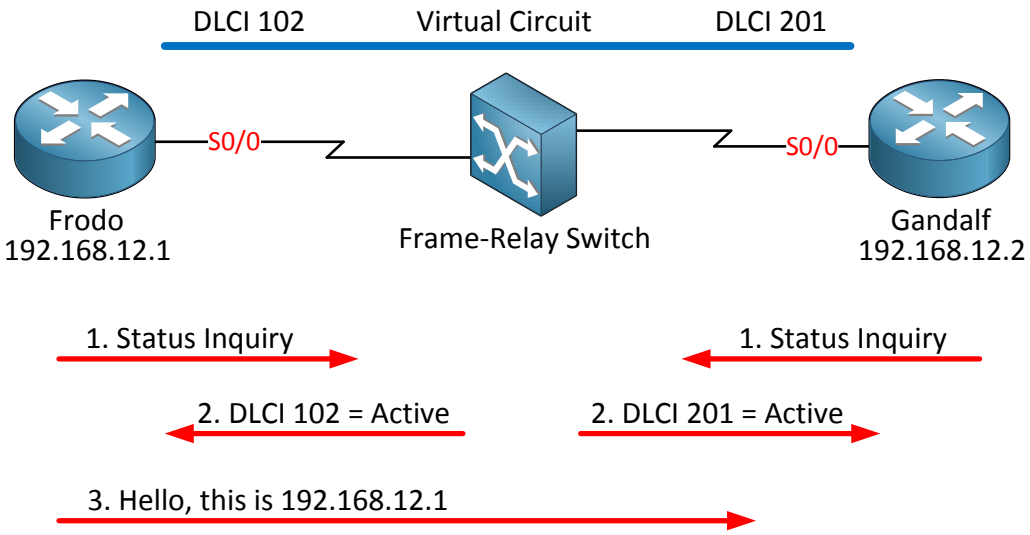


Router Frodo in my example above has mapped the IP address of router Gandalf (192.16.12.2) to its local DLCI 102. That's inverse ARP. Let's see it in more detail:

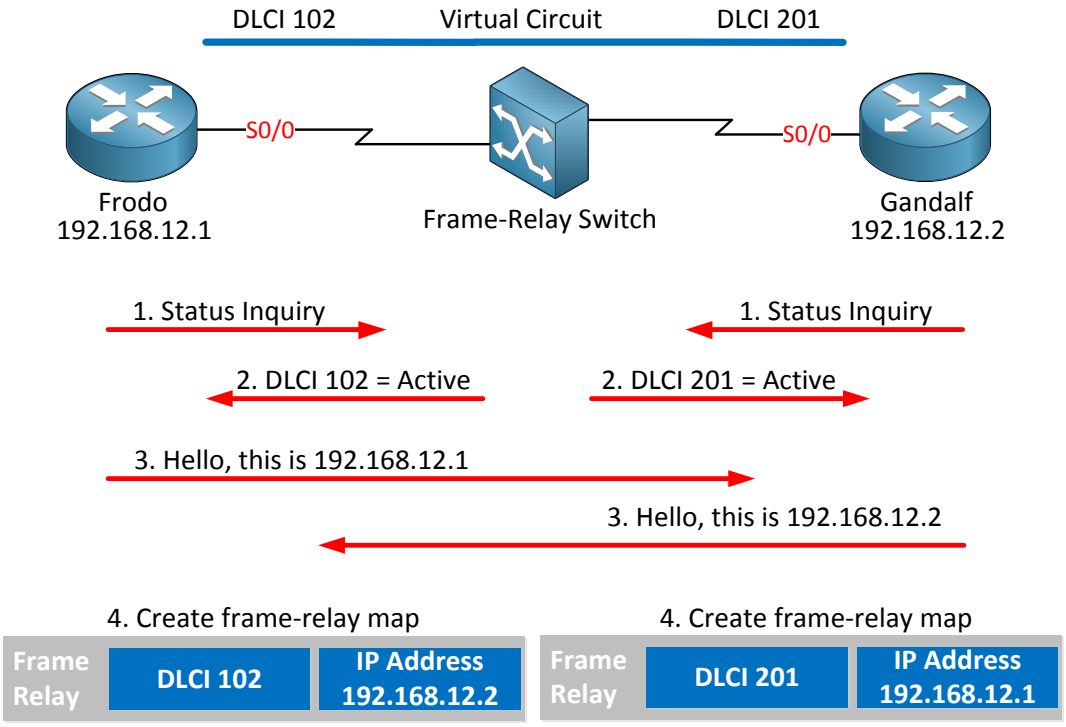


When we configure frame-relay here's what happens:

1. Our router will do a status enquiry using LMI.
2. The frame-relay switch will give us our DLCI number (or you can configure it yourself).

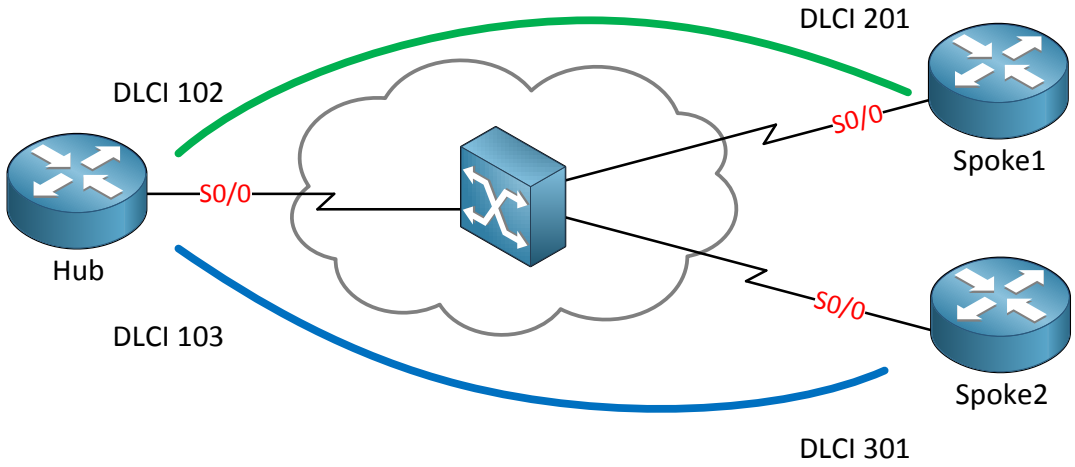


Once our routers know the PVC is active they will send a hello message with their IP address. In my example you only see router Frodo sending this but of course router Gandalf will also send its IP address.

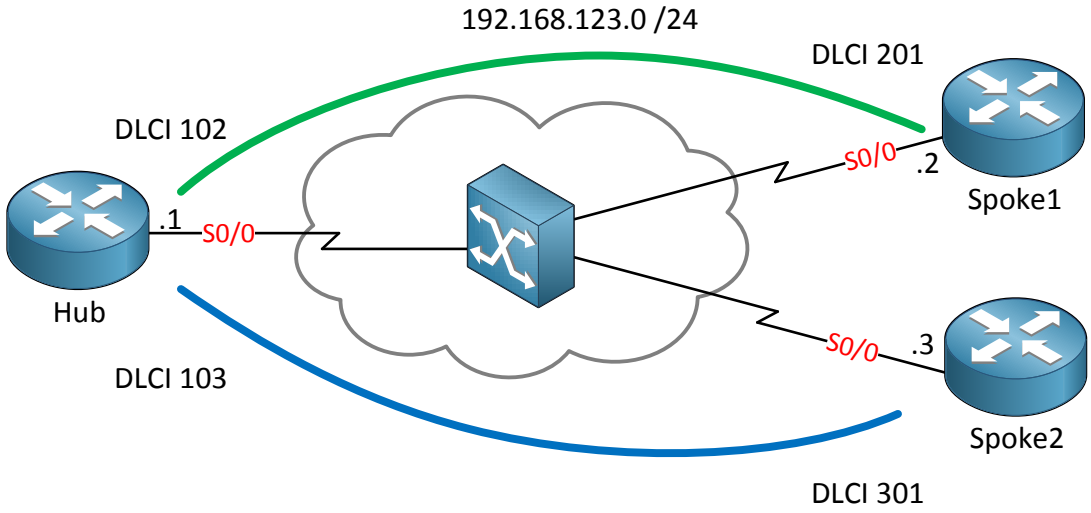


Router Frodo will now know it can reach IP address 192.168.12.2 by sending traffic through the PVC with DLCI 102. Router Gandalf will know that it can reach IP address 192.168.12.1 through the PVC with DLCI 201.

I hope this refreshes your frame-relay knowledge now I'm going to show you the different ways of configuring EIGRP over your frame-relay network.



The topology above is the one I'm going to use to show you how to run EIGRP over a frame-relay network. On the left side we have a hub router and on the ight side 2 spoke routers. This is a classic example of a hub and spoke model and ideal for a lab.



Same topology but I added the 192.168.123.0 /24 subnet. As you can see our Hub router has the .1 IP address, Spoke1 has .2 and Spoke2 has .3.

Since we are using a single IP subnet we are using frame-relay point-to-multipoint. Keep in mind that with frame-relay the physical interface is point-to-multipoint by default!

```
Hub(config)#interface s0/0
Hub(config-if)#ip address 192.168.123.1 255.255.255.0
Hub(config-if)#encapsulation frame-relay
```

```
Spoke1(config)#interface serial 0/0
Spoke1(config-if)#ip address 192.168.123.2 255.255.255.0
Spoke1(config-if)#encapsulation frame-relay
```

```
Spoke2(config)#interface serial 0/0
Spoke2(config-if)#ip address 192.168.123.3 255.255.255.0
```

```
Spoke2(config-if)#encapsulation frame-relay
```

This is the configuration of the interfaces of the three routers. Just a clean simple point-to-multipoint frame-relay configuration.

```
Hub(config)#router eigrp 123
Hub(config-router)#network 192.168.123.0
```

```
Spoke1(config)#router eigrp 123
Spoke1(config-router)#network 192.168.123.0
```

```
Spoke2(config)#router eigrp 123
Spoke2(config-router)#network 192.168.123.0
```

This is the EIGRP configuration of our routers. Not so hard right? Basic EIGRP router commands and you must change the serial interface to frame-relay encapsulation.

```
Hub#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 123: Neighbor 192.168.123.2 (Serial0/0) is
up: new adjacency
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 123: Neighbor 192.168.123.3 (Serial0/0) is
up: new adjacency
```

```
Spoke1#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 123: Neighbor 192.168.123.1 (Serial0/0) is
up: new adjacency
```

```
Spoke2#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 123: Neighbor 192.168.123.1 (Serial0/0) is
up: new adjacency
```

And we can see the EIGRP neighbor adjacencies are working.

```
Hub#show frame-relay map
Serial0/0 (up): ip 192.168.123.2 dlci 102(0x66,0x1860), dynamic,
              broadcast,, status defined, active
Serial0/0 (up): ip 192.168.123.3 dlci 103(0x67,0x1870), dynamic,
              broadcast,, status defined, active
```

Take a good look at the output of the **show frame-relay map** command on the hub router. Inverse ARP is enabled by default which is why you see the word **dynamic** in the frame-relay mappings. We see that it has learned about the IP addresses of the spoke routers and to which local DLCI numbers we have to map them.

There are 3 important things to learn in the output above:

- We didn't configure any frame-relay mappings which means that Inverse ARP is enabled by default.
- The word **dynamic** means we are using Inverse ARP.
- The keyword **broadcast** means its emulating broadcast traffic so we can use multicast as well. Inverse ARP enabled this for us.

Let's disable Inverse ARP on the hub router and configure some frame-relay mappings ourselves:

```
Hub(config)#interface serial 0/0
Hub(config-if)#no frame-relay inverse-arp
```

That's all there is. Type in **no frame-relay inverse-arp** under the interface and you'll disable Inverse ARP.

```
Hub#clear frame-relay inarp
```

Don't forget to remove the pre-learnt frame-relay mappings by using the **clear frame-relay inarp** command.

Next step is to configure the frame-relay mappings ourselves:

```
Hub(config)#interface serial 0/0
Hub(config-if)#frame-relay map ip 192.168.123.2 102 broadcast
Hub(config-if)#frame-relay map ip 192.168.123.3 103 broadcast
```

You configure it on the interface level by using the **frame-relay map** command.

You have to specify the **broadcast** keyword or your frame-relay network will only support unicast! Since EIGRP uses multicast you'll run into trouble if you forget to use this.

What else can we do with EIGRP and frame-relay? Let's forget about the physical interface and move our frame-relay commands to a sub-interface!

```
Hub(config)#interface s0/0
Hub(config-if)#no frame-relay map ip 192.168.123.3 103 broadcast
Hub(config-if)#no frame-relay map ip 192.168.123.2 102 broadcast
Hub(config-if)#no ip address 192.168.123.1 255.255.255.0
```

```
Hub(config)#interface s0/0.123 multipoint
Hub(config-subif)#ip address 192.168.123.1 255.255.255.0
Hub(config-subif)#frame-relay map ip 192.168.123.2 102 broadcast
Hub(config-subif)#frame-relay map ip 192.168.123.3 103 broadcast
```

I'll move the IP address and the frame-relay mappings from the physical interface to a sub-interface.

```
Hub#show running-config | begin interface Serial0/0
interface Serial0/0
 no ip address
 encapsulation frame-relay
 serial restart-delay 0
 no frame-relay inverse-arp
!
interface Serial0/0.123 multipoint
 ip address 192.168.123.1 255.255.255.0
 frame-relay map ip 192.168.123.2 102 broadcast
 frame-relay map ip 192.168.123.3 103 broadcast
```

Here's an overview of the physical and sub interface.

There are only two things I'm leaving on the physical interface:

- The encapsulation type, we want to run frame-relay right?
- Disable Inverse ARP with the no frame-relay inverse-arp command.

We don't want any frame-relay mappings on the physical interface since we need them on the sub-interface. If you create a sub-interface you need to specify whether it's a multipoint or point-to-point. I'm still using a single IP subnet for all routers so we are using multipoint in this situation.

```
Hub(config)#interface serial 0/0.123
Hub(config-subif)#no ip split-horizon eigrp 123
```

This is an important step you should not forget. On a physical interface split-horizon is **disabled** but on the sub-interface it is **enabled** by default! Don't forget to add your frame-relay mappings on the sub-interface with the broadcast keyword at the end.

```
Hub#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 123: Neighbor 192.168.123.3 (Serial0/0.123)
is resync: split horizon changed
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 123: Neighbor 192.168.123.2 (Serial0/0.123)
is resync: split horizon changed
```

You'll see a quick message that split horizon has changed.

What happens if I forget the broadcast keyword in the frame-relay mapping? Like the following example:

```
Hub(config-subif)#no frame-relay map ip 192.168.123.2 102 broadcast
Hub(config-subif)#no frame-relay map ip 192.168.123.3 103 broadcast
Hub(config-subif)#frame-relay map ip 192.168.123.2 102
Hub(config-subif)#frame-relay map ip 192.168.123.3 103
```

Our frame-relay configuration doesn't support broadcast nor multicast anymore…only unicast.

```
Hub#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 123: Neighbor 192.168.123.3 (Serial0/0.123)
is down: Interface Goodbye received
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 123: Neighbor 192.168.123.2 (Serial0/0.123)
is down: Interface Goodbye received
```

After a short time you'll see messages like this.
Is there still any method to get EIGRP working? Sure there is! Just use the **neighbor** command. If you manually specify EIGRP neighbors we will switch over to unicast.

```
Hub(config-subif)#router eigrp 123
Hub(config-router)#neighbor 192.168.123.3 serial 0/0.123
Hub(config-router)#neighbor 192.168.123.3 serial 0/0.123
```

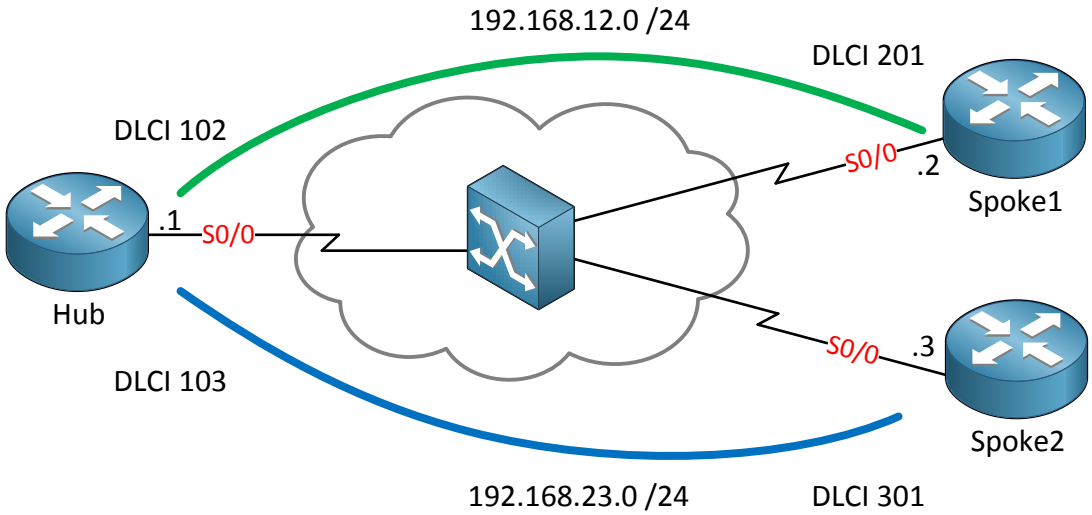
```
Spoke1(config)#router eigrp 123
Spoke1(config-router)#neighbor 192.168.123.1 serial 0/0
```

```
Spoke2(config)#router eigrp 123
Spoke2(config-router)#neighbor 192.168.123.1 serial 0/0
```

Here we go…specify the EIGRP neighbors and we no longer need multicast and you can forget about the broadcast keyword in your frame-relay mappings.

Those are all the options we have for running EIGRP over frame-relay using the **multipoint** physical or sub-interface.

I still need to show you the **point-to-point** sub-interface method:



We are using the same topology to demonstrate point-to-point EIGRP over frame-relay with one difference. Point-to-point links require an **IP Subnet per PVC**:

- Hub and Spoke1: 192.168.12.0 /24
- Hub and Spoke2: 192.168.13.0 /24

```
Hub(config)#interface serial 0/0
Hub(config-if)#encapsulation frame-relay

Hub(config)#interface serial 0/0.1 point-to-point
Hub(config-subif)#ip address 192.168.12.1 255.255.255.0
Hub(config-subif)#frame-relay interface-dlci 102

Hub(config)#interface serial 0/0.2 point-to-point
Hub(config-subif)#ip address 192.168.13.1 255.255.255.0
Hub(config-subif)#frame-relay interface-dlci 103
```

```
Spoke1(config)#interface s0/0
Spoke1(config-if)#encapsulation frame-relay
Spoke1(config-if)#exit
```

```
Spoke1(config)#interface serial 0/0.1 point-to-point
Spoke1(config-subif)#ip address 192.168.12.2 255.255.255.0
Spoke1(config-subif)#frame-relay interface-dlci 201
```

```
Spoke2(config)#interface serial 0/0
Spoke2(config-if)#encapsulation frame-relay
Spoke2(config-if)#exit
Spoke2(config)#interface serial 0/0.1 point-to-point
Spoke2(config-subif)#ip address 192.168.13.3 255.255.255.0
Spoke2(config-subif)#frame-relay interface-dlci 301
```

Here is the configuration for the hub and spoke routers. You only have to specify encapsulation frame-relay on the physical interface. The rest of the commands are on the sub-interfaces. Your router can't read your mind and find out on which sub-interfaces which DLCI's should be so you have to configure it yourself. We don't use the frame-relay map command for point-to-point sub-interfaces but you have to use the **frame-relay interface-dlci** command here.

Since we are using 2 sub-interfaces on the hub router instead of a single multipoint interface we don't have to deal with pesky split-horizon issues here!

Now you have seen all the different ways of configuring EIGRP over frame-relay networks. There is one thing left you have to understand when running EIGRP over frame-relay networks and this is about bandwidth utilization.

- By default EIGRP can use up to **50% of interface bandwidth for EIGRP traffic**.
- The percentage can be changed.
- Point-to-point interfaces:
  - Bandwidth **treated as T1 interface** (1.544Mbit).
  - Configure the bandwidth yourself to reflect the true bandwidth.
- Multipoint interfaces:
  - Bandwidth on physical interface is **divided by the number of neighbors**.

What is the problem here? Let's say your frame-relay provider has given you a PVC with a CIR of 64kbps. Your point-to-point interface is by default treated as T1 which is 1.544Mbit. EIGRP can take 50% of the bandwidth….50% of 1.544Mbit is 768kbps and voila…your PVC is flooded with EIGRP packets!

You can solve this by setting the correct bandwidth using the **bandwidth** command on the interface and/or by changing the percentage that EIGRP may use for traffic.

```
Hub(config)#interface s0/0.1 point-to-point
Hub(config-subif)#band
Hub(config-subif)#bandwidth 64
```

This is how you change the bandwidth on the sub-interface. If you change the bandwidth on the physical interface your sub-interface **will not inherit** the bandwidth!

You need to specify it on each sub-interface yourself.

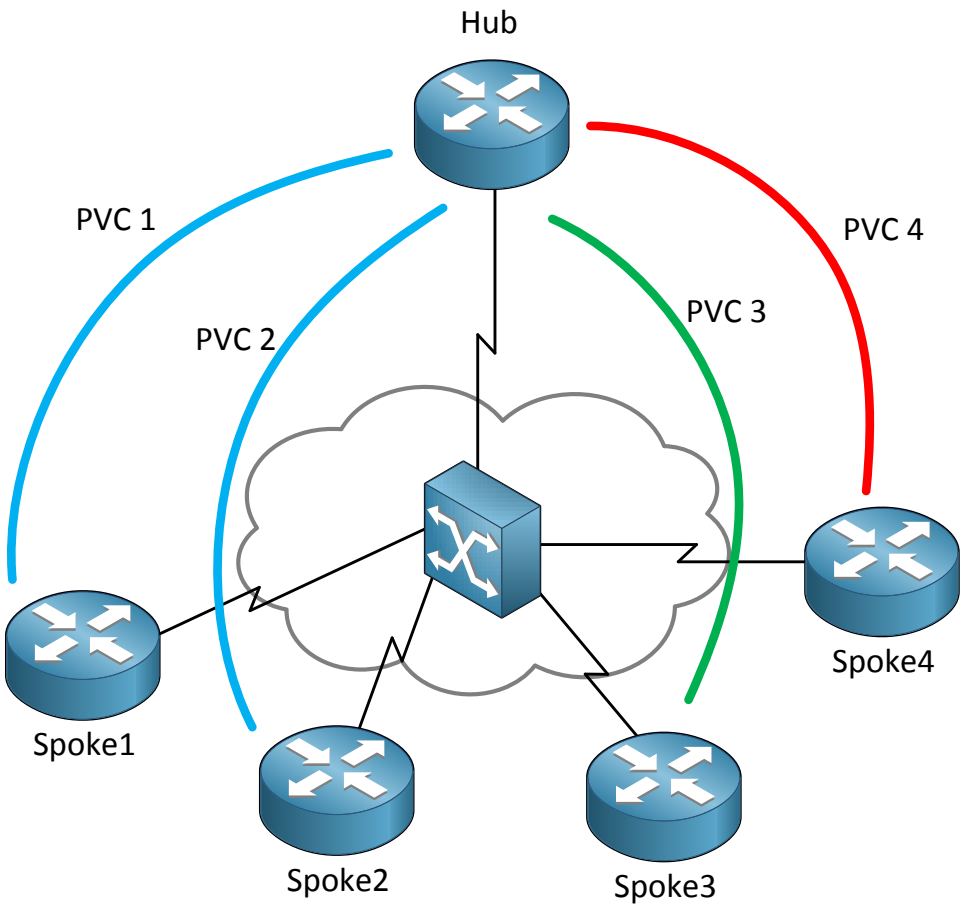
```
Hub#show interfaces serial 0/0.1
Serial0/0.1 is up, line protocol is up
  Hardware is M4T
  Internet address is 192.168.12.1/24
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec,
```

You can see the new bandwidth by using the **show interface** command.

```
Hub(config)#interface s0/0.1 point-to-point
Hub(config-subif)#ip bandwidth-percent eigrp 123 25
```

By using the **ip bandwidth-percent eigrp** command you can change the percentage EIGRP may use for traffic. In my example I set it to 25% for EIGRP AS 123. Why would I want to change this percentage? Keep in mind changing the bandwidth also influences the metric so if you changed the bandwidth because of routing policy you might not want to change it. Imagine you have a FastEthernet link but set the bandwidth to 64kbps so the interface is less attractive to EIGRP. 50% of 64kbps is only 32kbps that EIGRP allows itself to use on this FastEthernet interface...that's not a lot right? That's why you can set the bandwidth percentage to above 100%.

Are you following me so far? Good! Let's crank it up a notch by looking at some funky examples with EIGRP bandwidth related issues.



In the example above I have 5 routers in a hub and spoke model. The router on top is our hub and I have 4 spoke routers at the bottom. For each spoke there is a PVC and they each have a different CIR:

- PVC 1: CIR 128kbps
- PVC 2: CIR 128kbps
- PVC 3: CIR 256kbps
- PVC 4: CIR 64kbps

If we configure this frame-relay network as multipoint we might run into an issue. What happens when the spoke3 router sends EIGRP updates at 50% of its capacity meant for the spoke4 router?

50% of 256kbps = 128kbps

PVC4 only has a CIR of 64kbps so it will be overloaded with EIGRP traffic…not a good idea.

How can we fix this? The best method is to get rid of the multipoint setup and use point-to-point sub-interfaces since it allows you to set the bandwidth per sub-interface and thus per neighbor. Are we going to do the easy solution? Of course not ;)
If you want to keep the multipoint setup this is what you need to do:

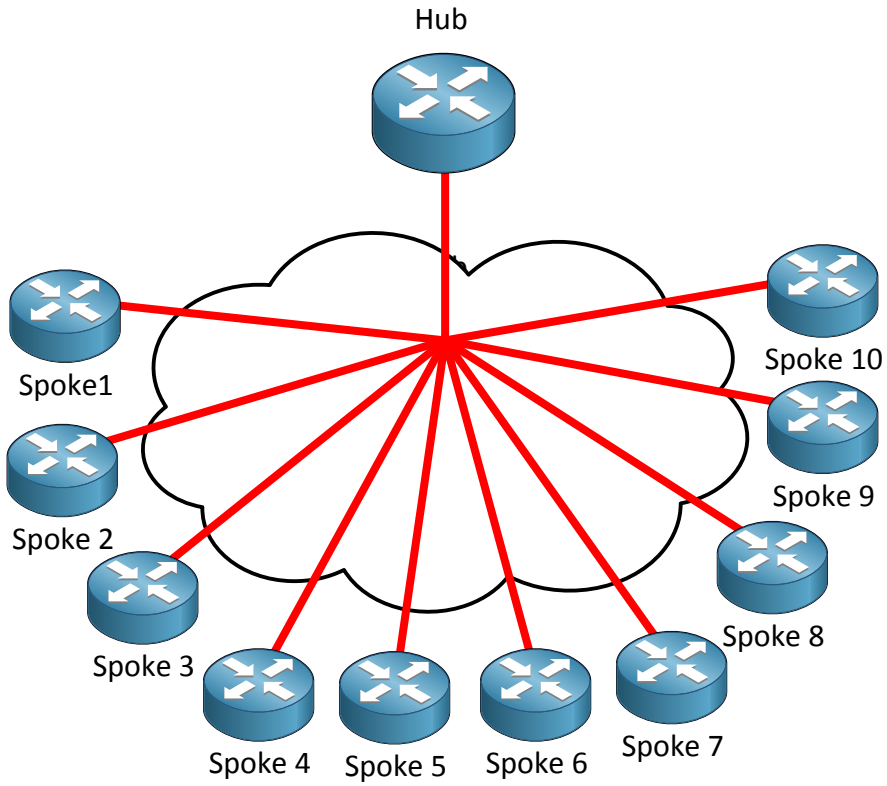
- Take the PVC with the lowest CIR. In our example this is PVC4 with a CIR of 64kbps.
- Multiply 64kbps with the number of PVCs and configure this as the bandwidth on the hub router.

```
Hub(config)#interface serial 0/0
Hub(config-if)#bandwidth 256
```

If you use multipoint interfaces EIGRP will divide the bandwidth over the number of neighbors. In our example this means each PVC will get 64kbps which is the bandwidth of the lowest bandwidth PVC.

This solution will not get the maximum out of the PVCs with a higher CIR but will ensure that the lower CIR PVCs are not overburdened with traffic.

Let’s look at another topology:

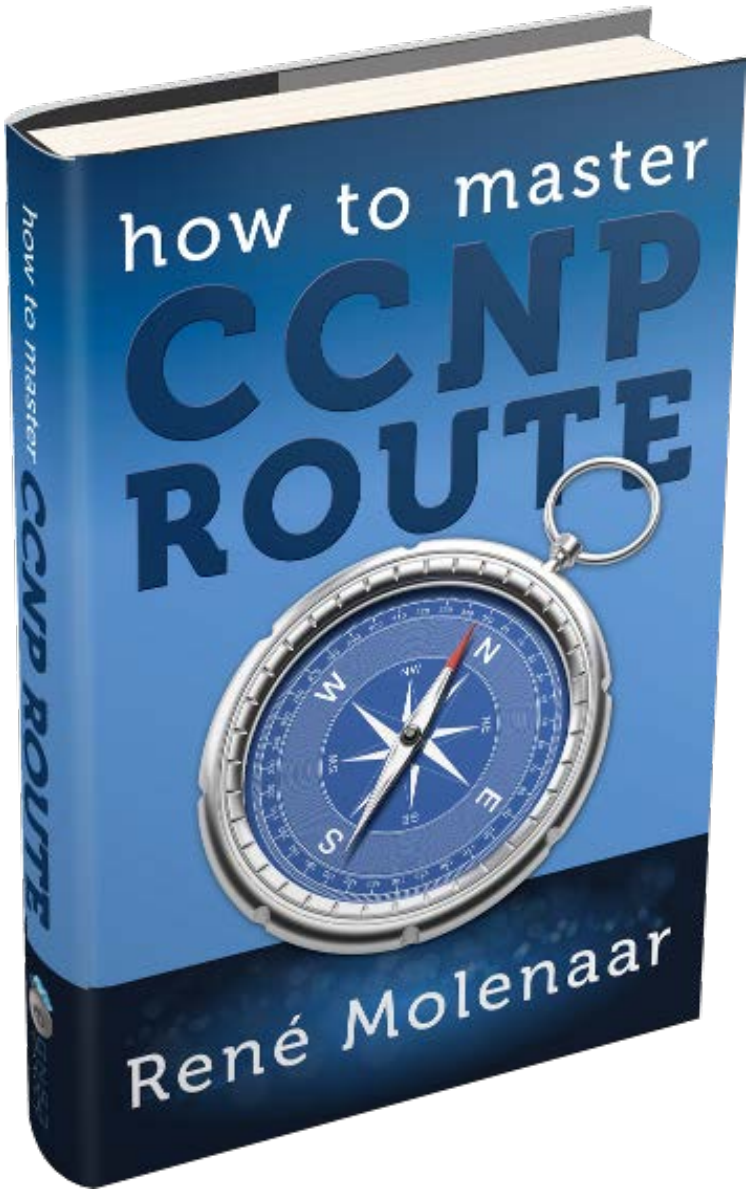


In the topology above I have one hub router and 10 spoke routers. All the PVCs are configured as point-to-point and have a CIR of 64kbps. The physical interface at the hub router only has a bandwidth of 256kbps and has been configured correctly.

Do you enjoy reading this sample of How to Master CCNP ROUTE ?

Click on the link below to get the full version.

Get How to Master CCNP ROUTE Today



```
Hub(config)#interface serial 0/0
Hub(config-if)#bandwidth 256
Hub(config-if)#exit

Hub(config)#interface serial 0/0.1 point-to-point
Hub(config-subif)#ip address 192.168.12.1 255.255.255.0
Hub(config-subif)#frame-relay interface-dlci 102

Hub(config)#interface serial 0/0.2 point-to-point
Hub(config-subif)#ip address 192.168.13.1 255.255.255.0
Hub(config-subif)#frame-relay interface-dlci 103


Hub(config)#interface serial 0/0.3 point-to-point
Hub(config-subif)#ip address 192.168.14.1 255.255.255.0
Hub(config-subif)#frame-relay interface-dlci 104
```

Here's part of the configuration of the hub router, I'm only showing the first 3 spoke routers. You can see the bandwidth has been configured correctly on the physical interface to 256kbps. Each PVC is configured as a point-to-point sub-interface.

What happens when our hub router tries to communicate with all spoke routers at the same time at full capacity? 10x 64kbps is 640kbps and our physical interface doesn't go faster than 256kbps…we'll run out of capacity! We have to make some changes to solve this problem.

```
Hub(config)#interface serial 0/0
Hub(config-if)#bandwidth 256
Hub(config-if)#exit

Hub(config)#interface serial 0/0.1 point-to-point
Hub(config-subif)#ip bandwidth-percent eigrp 123 128

Hub(config)#interface serial 0/0.2 point-to-point
Hub(config-subif)#ip bandwidth-percent eigrp 123 128

Hub(config)#interface serial 0/0.3 point-to-point
Hub(config-subif)#ip bandwidth-percent eigrp 123 128
```

We'll start with the hub router. The physical bandwidth of the interface is 256kbps and since we have 10 PVCs this bandwidth will be divided automatically between the PVCs, no need to configure the bandwidth **manually** on each point-to-point sub-interface.

256 / 10 = 25kbps for each PVC.

Each PVC has a bandwidth of 64kbps however so we need to do something to make sure EIGRP can still use 50% of the available bandwidth.

I have changed the percentage EIGRP AS 123 can use to 128%.

128% of 25kbps = 32kbps

32kbps is exactly 50% of the available CIR rate for each PVC.

What about the spoke routers? We need to change their configuration as well so they reflect the hub router. Let me show you one of them:

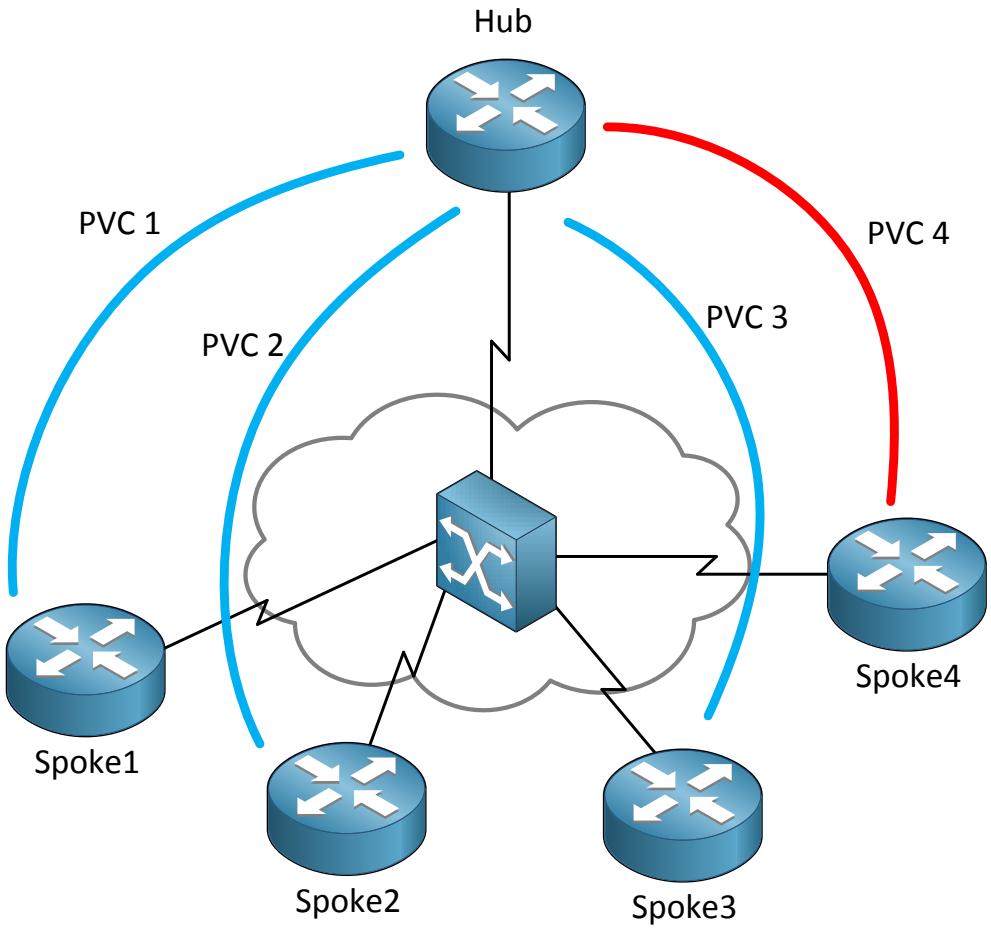
```
Spoke1(config)#interface serial 0/0
Spoke1(config-if)#bandwidth 25
Spoke1(config-if)#exit

Spoke1(config)#interface serial 0/0.1 point-to-point
Spoke1(config-subif)#ip bandwidth-percent eigrp 123 128
```

We set the bandwidth at 25kbps and the percentage EIGRP can use at 128% for AS 123.

128% of 25kbps = 32kbps

32kbps is exactly 50% of the available CIR rate of our PVC.
There is one downside to this solution. Since EIGRP divides the bandwidth of multipoint interfaces automatically over the number of neighbors there shouldn't be any changes in the number of EIGRP neighbors or our calculation is incorrect.

Almost there…I promise! Now you have seen the multipoint and point-to-point solution but there's also a **hybrid** option. We are going to mix the multipoint and point-to-point interfaces for this one!



In the example above I have the same topology as our first example. The router on top is our hub and I have 4 spoke routers at the bottom. There's something different with the CIR of our PVCs:

- PVC 1,2 and 3: CIR 256kbps
- PVC 4: CIR 128kbps

So PVC 1, 2 and 3 have the same CIR rate of 256kbps while PVC4 is slower at 128kbps. We can combine the 3 PVCs with the 256kbps CIR on a multipoint interface while the slow PVC is configured on a separate point-to-point interface.

```
Hub(config)#interface serial 0/0.123 multipoint
Hub(config-subif)#bandwidth 768
Hub(config-subif)#exit

Hub(config)#interface serial 0/0.1 point-to-point
Hub(config-subif)#bandwidth 128
```

This is what we have to do on the hub router. PVC1, 2 and 3 we are going to configure under the multipoint sub-interface and give it a bandwidth of 768kbps. EIGRP automatically divides the bandwidth over the number of EIGRP neighbors so each of them gets 256kbps which matches the CIR rate of 256kbps.

The slower PVC number 4 will be configured as a separate point-to-point sub-interface where we set the correct bandwidth, that's it!

We made it to the end of this chapter. I believe EIGRP with frame-relay and the different bandwidth/CIR configurations is probably the most difficult part to understand about this subject. If you want some good practice check out the frame-relay labs I created for EIGRP.

If you are a little fuzzy on frame-relay you can start with this lab:

http://gns3vault.com/Frame-Relay/frame-relay-basics.html

Next stop is to configure EIGRP over frame-relay:

http://gns3vault.com/EIGRP/eigrp-over-frame-relay-with-sub-interfaces.html

http://gns3vault.com/EIGRP/eigrp-over-frame-relay-with-multipoint-interface.html

Once you feel familiar with EIGRP over frame-relay you can work on the different bandwidth pacing labs:

http://gns3vault.com/EIGRP/eigrp-multipoint-bandwidth-pacing.html

http://gns3vault.com/EIGRP/eigrp-point-to-point-bandwidth-pacing.html

http://gns3vault.com/EIGRP/eigrp-hybrid-bandwidth-pacing.html

5. EIGRP Authentication

Routing protocols can be configured to prevent receiving false routing updates and EIGRP is no exception. If you don't use authentication and you are running EIGRP someone could try to form an EIGRP neighbor adjacency with one of your routers and try to mess with your network…we don't want that to happen right?

What options do we have to authenticate EIGRP?

- MD5 authentication.

EIGRP only offers MD5 authentication, there's no plaintext authentication.

What does authentication offer us?

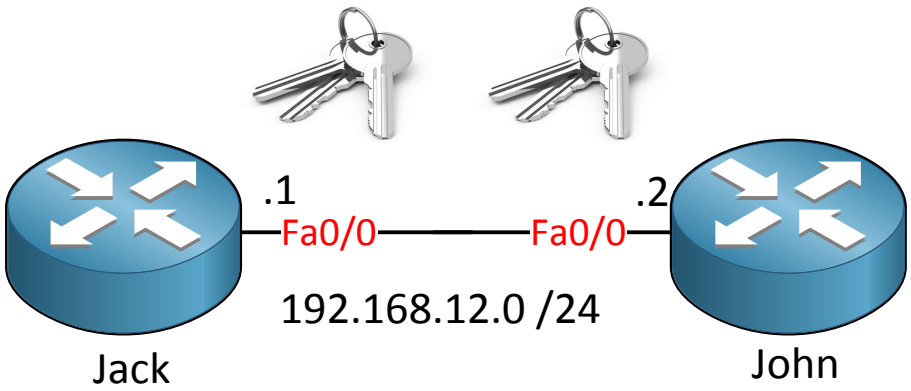
- Your router will authenticate the source of each routing update packet that it will receive.
- Prevents false routing updates from sources that are not approved.
- Ignore malicious routing updates.

A potential hacker could be sitting on your network with a laptop running GNS3 / Dynamips, boot up a Cisco router and try the following things:

- Try to establish a neighbor adjacency with one of your routers and advertise junk routes.
- Send malicious packets and see if you can drop the neighbor adjacency of one of your authorized routers.

In order to configure EIGRP authentication we need to do the following:

- Configure a key-chain
    - Configure a key ID under the key-chain.
        - Specify a password for the key ID.
        - Optional: specify accept and expire lifetime for the key.



I'm going to use router Jack and John again and this time we will configure MD5

authentication for EIGRP.
The configuration for both routers is very basic:

```
Jack(config)#interface fastEthernet 0/0
Jack(config-if)#ip address 192.168.12.1 255.255.255.0

Jack(config)#router eigrp 12
Jack(config-router)#network 192.168.12.0
```

```
John(config)#interface fastEthernet 0/0
John(config-if)#ip address 192.168.12.2 255.255.255.0

John(config)#router eigrp 12
John(config-router)#network 192.168.12.0
```

Key ID 1
String: Banana

Keychain "KingKong"

We'll start by specifying a keychain. I called mine "KingKong" but it can be different on both routers, it doesn't matter. The Key ID is a value that has to match on both routers and the key-string is the password which has to match of course.

```
Jack(config)#key chain KingKong
Jack(config-keychain)#key 1
Jack(config-keychain-key)#key-string Banana
```

```
Jack(config)#interface fastEthernet 0/0
Jack(config-if)#ip authentication mode eigrp 12 md5
Jack(config-if)#ip authentication key-chain eigrp 12 KingKong
```

First you have to create the keychain and then you need to activate it on the interface. The "12" is the AS number of EIGRP. The configuration on router John is exactly the same.

```
John#debug eigrp packets
EIGRP Packets debugging is on
    (UPDATE, REQUEST, QUERY, REPLY, HELLO, IPXSAP, PROBE, ACK, STUB,
SIAQUERY, SIAREPLY)
```

```
John# EIGRP: FastEthernet0/0: ignored packet from 192.168.12.1, opcode = 5
(authentication off or key-chain missing)
```

You can check if your configuration is correct by using **debug eigrp packets**. You can see that we received a packet with MD5 authentication but I didn't enable MD5 authentication

yet on router John.

Let's fix it:

```
John(config)#key chain KingKong
John(config-keychain)#key 1
John(config-keychain-key)#key-string Banana

John(config)#interface fastEthernet 0/0
John(config-if)#ip authentication mode eigrp 12 md5
John(config-if)#ip authentication key-chain eigrp 12 KingKong
```

Right away I can see that the EIGRP neighbor adjacency is working:

```
John# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.1
(FastEthernet0/0) is up: new adjacency
```

What if I entered a wrong key-string?

```
Jack(config)#key chain KingKong
Jack(config-keychain)#key 1
Jack(config-keychain-key)#key-string Apples
```

Let's see if KingKong likes apples…

```
John# EIGRP: pkt key id = 1, authentication mismatch
```

You will see the message above in the debug output on router John. At least it tells us that key 1 is the one with the error.

If you want to spice it up a bit you can set an **accept** and **expire** lifetime on keys. The idea behind this is that you can have keys that are only valid for a day, a week, a month or something else. Do you want to use this in real life? It might enhance security but it also make maintenance a bit more complex…

Before you configure keys with a limited lifetime make sure you set the correct time and date. You can do this manually on each router but it's better to use a NTP (Network Time Protocol) server so all the routers have the same time/date.

See if you can configure authentication with this lab:

http://gns3vault.com/EIGRP/eigrp-authentication-rotating-key.html

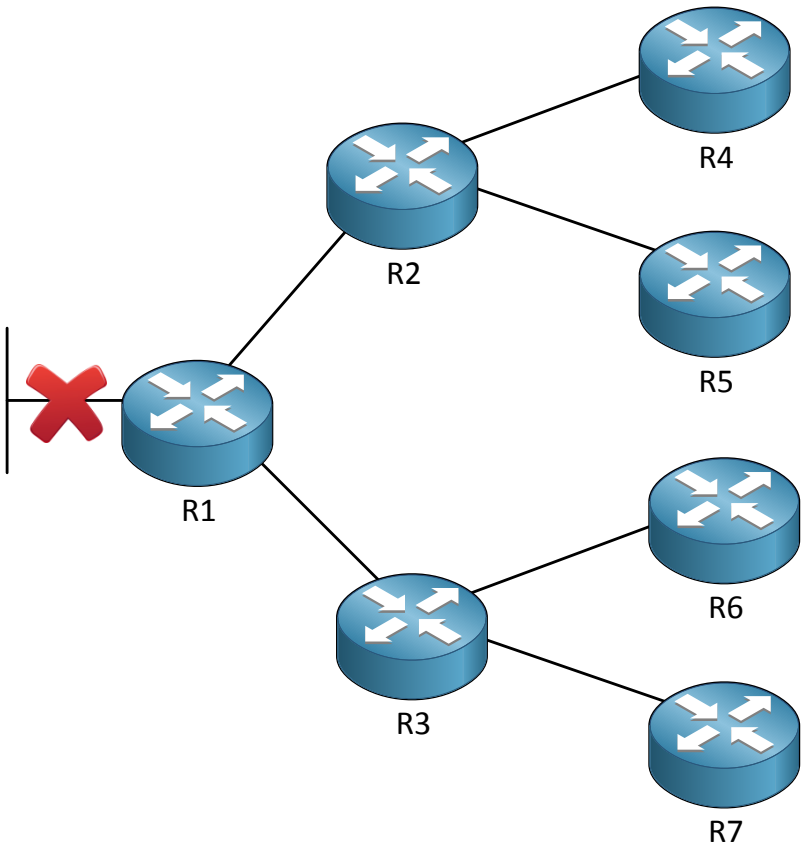
6. EIGRP Advanced Features

This is our final EIGRP chapter and we will look at some more advanced EIGRP features. We will dive a little bit more into the EIGRP query process and how stub routers can help us solve some problems.

EIGRP is designed for large enterprise networks but having one big EIGRP network (5000+ prefixes and many hops) can lead to some problems:

- Lots of EIGRP prefixes equal a large topology table and routing table.
- Calculating the successor router will take longer if you have many EIGRP neighbors and different paths.
- If there are many backup paths EIGRP will have to see if there are 1 or more feasible successors, this will take longer.
- More information means our EIGRP routers have to work harder to process everything.
- When EIGRP loses a route and there is no feasible successor the route will go from **passive** to **active** and the router starts sending queries to its neighbors.
- EIGRP sends queries on all interfaces except the interface of the successor.

We have talked about summarization before and how it helps to reduce the size of the routing table and stops the query process. Let me describe the EIGRP query process in detail for you:

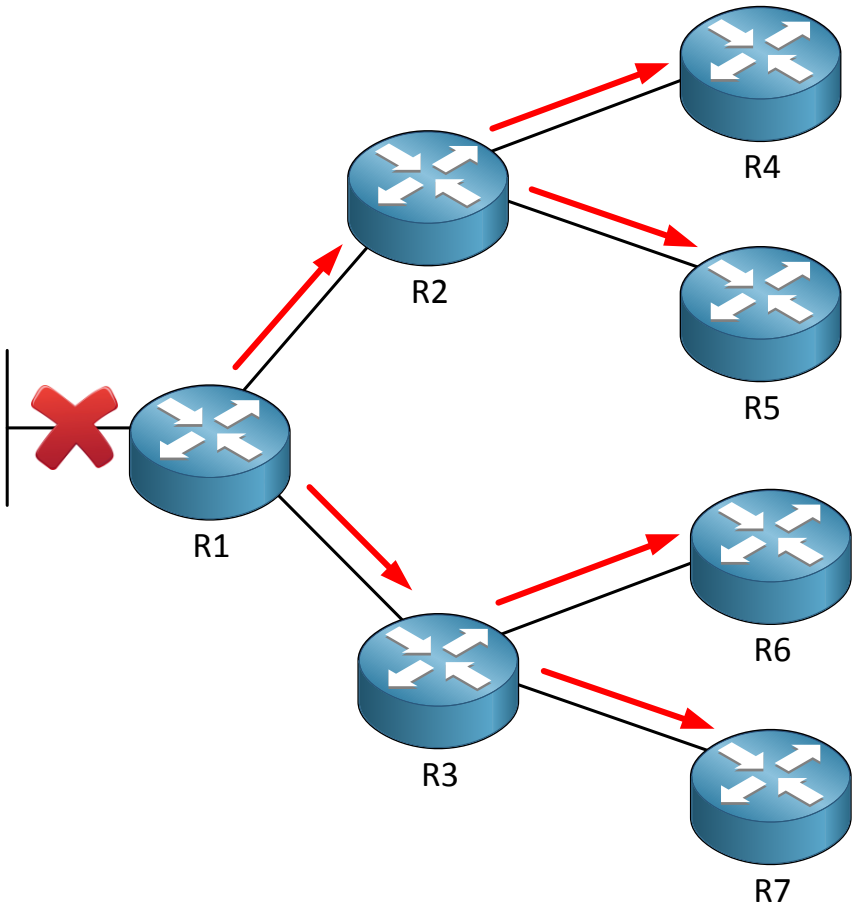


In the topology above we are running EIGRP on all of the routers. Router 1 has a link failure

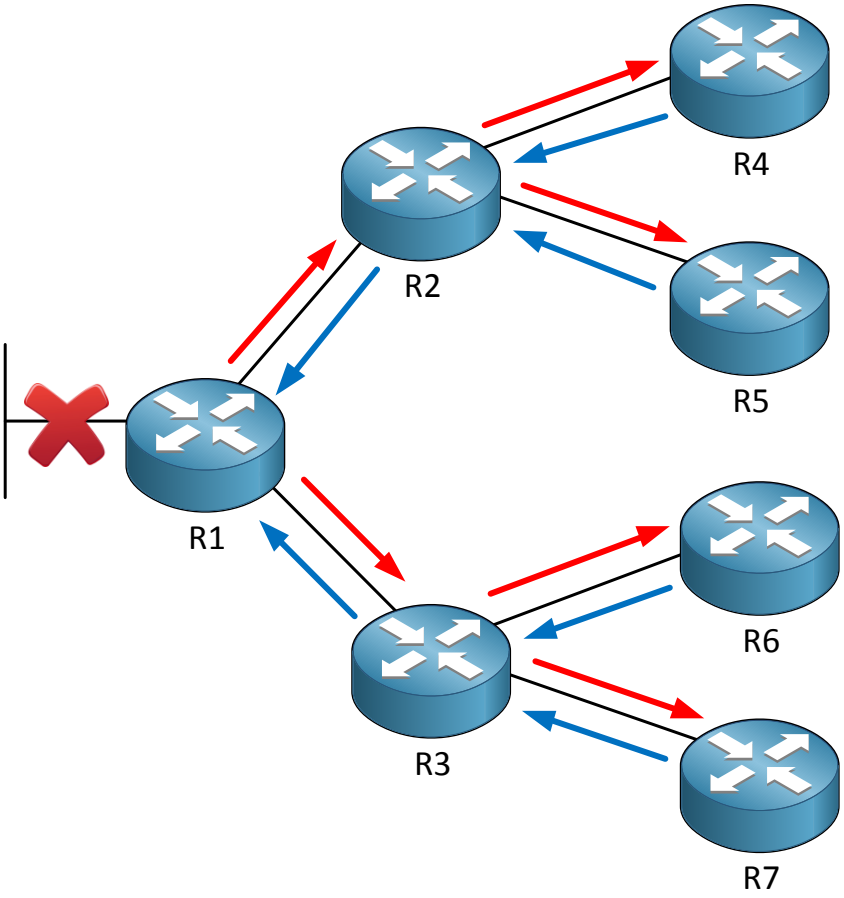
and as a result has lost its successor route to a particular network on the left side. There is no feasible successor so the route is going from passive to active and we will send a query to router 2 and 3.

There are 2 things that can happen at this moment:

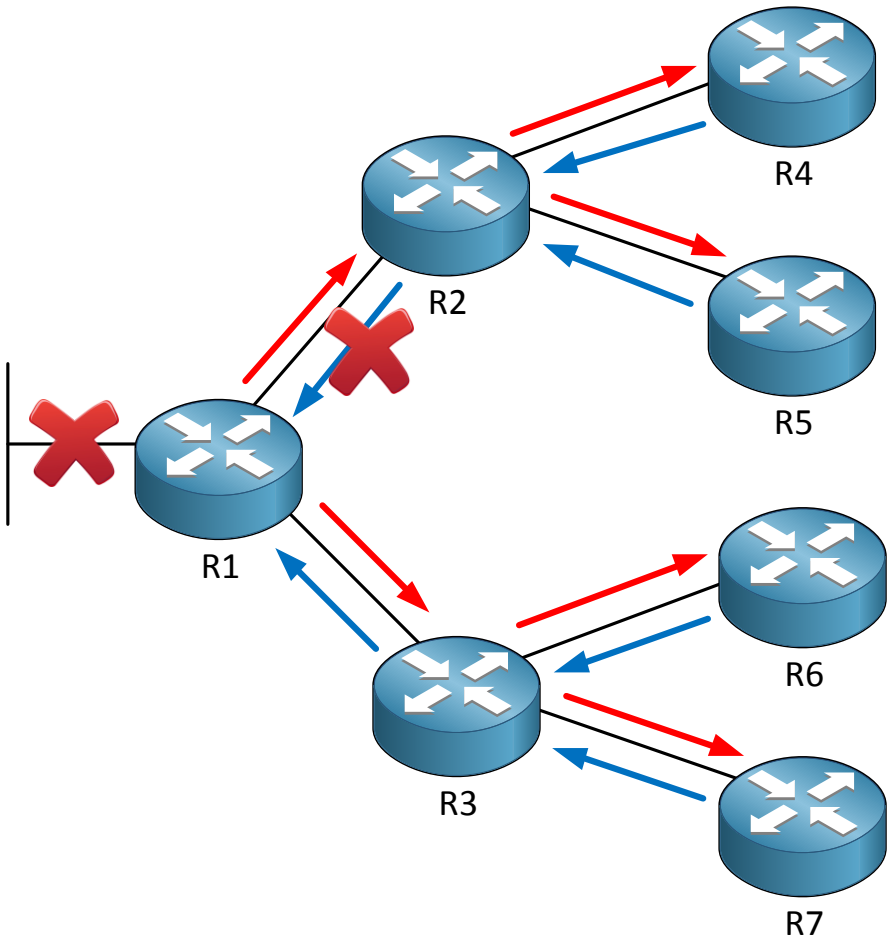
- Router 2 or 3 has information about this particular route and will send information about it to router 1. The query process is now over.
- Router 2 or 3 don't know anything about this route and will send a query themselves to their neighbors router 4, 5 and router 6 and 7. Router 2 or 3 will not send a reply to router 1 until they heard a response from all their neighbors.



In our topology nobody has a clue which network router 1 is looking for. They will forward their queries to their own neighbors. The red arrows indicate the query packet.



There are no other neighbors behind router 4, 5, 6 or 7. They will send a reply to router 2 and 3 to let them know they don't know the answer. Router 2 and 3 will send a reply to router 1 to tell them they are sorry but this is it. That's a lot of packets for just one route that was lost right?

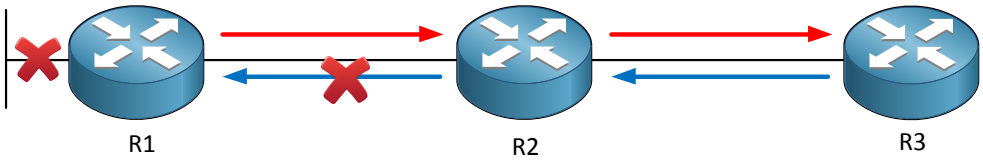


Let's make things even more interesting. Look at my picture above and you'll see that the reply from router 2 never makes it back to router 1. EIGRP is a reliable protocol and for each query a router sends to its neighbors it must **get a reply in response within 3 minutes**. If the router does not receive a reply to ALL its outstanding queries it will put the route in **SIA (Stuck in Active)** state and will kill the neighbor adjacency. By dropping the neighbor adjacency you will lose all the routes you learned from this neighbor which means the router will start sending queries for all those routes as well. Not a pretty sight right?

How is it possible that a reply never makes it back?

- The router that gets the query is too busy because of memory problems or a CPU that's too busy. It might not get the chance to process the incoming query or send a reply packet.
- There are problems with the link between the neighbors so not all packets arrive.
- You have a unidirectional link failure so packets only flow in one direction. This can happen with fiber links.

Since IOS 12.1 Cisco decided to change the stuck in active process to reduce the number of unwanted lost neighbor adjacencies. They introduced two new packets called **SIA query** and **SIA reply**.



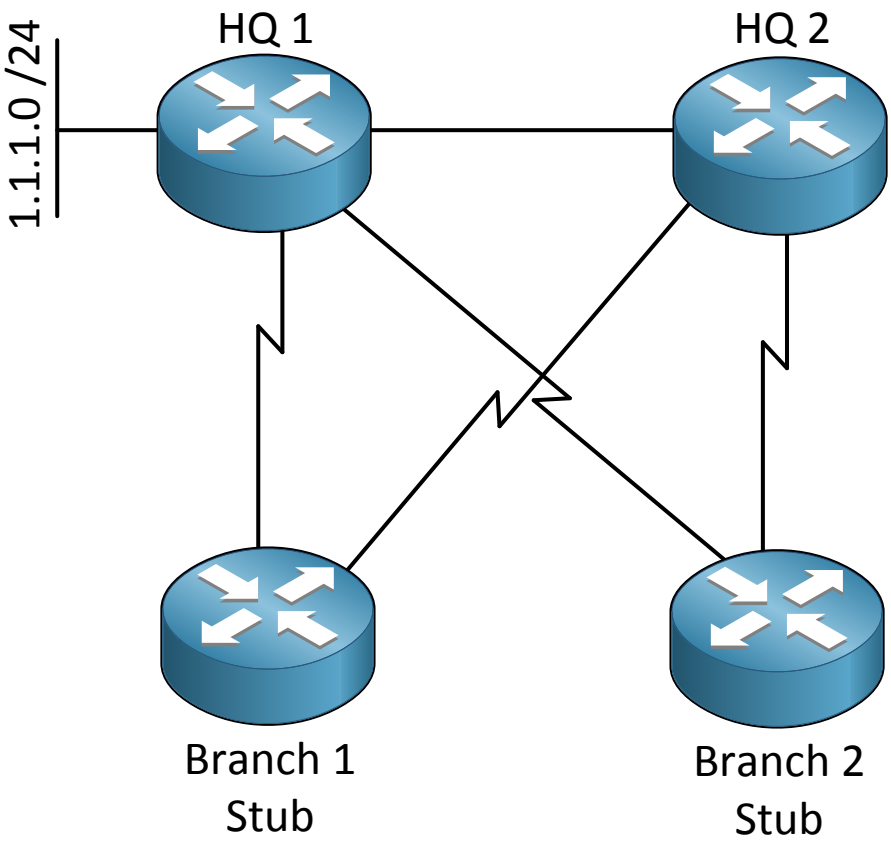
Before Cisco introduced SIA query and SIA reply this would happen:

1. Router 1 loses information about a network and has no feasible successor.
2. Router 1 sends a query to router 2.
3. Router 2 doesn't know the answer so sends a query as well to router 3.
4. Router 3 doesn't know the answer and sends a reply to router 2.
5. Router 2 sends a reply to router 1 to let him know he has no clue about this network.
6. Because of congestion the reply from router 2 never makes it back to router 1.
7. After 3 minutes router 1 will drop the neighbor adjacency with router 2 including all the routes it learnt from router 2.

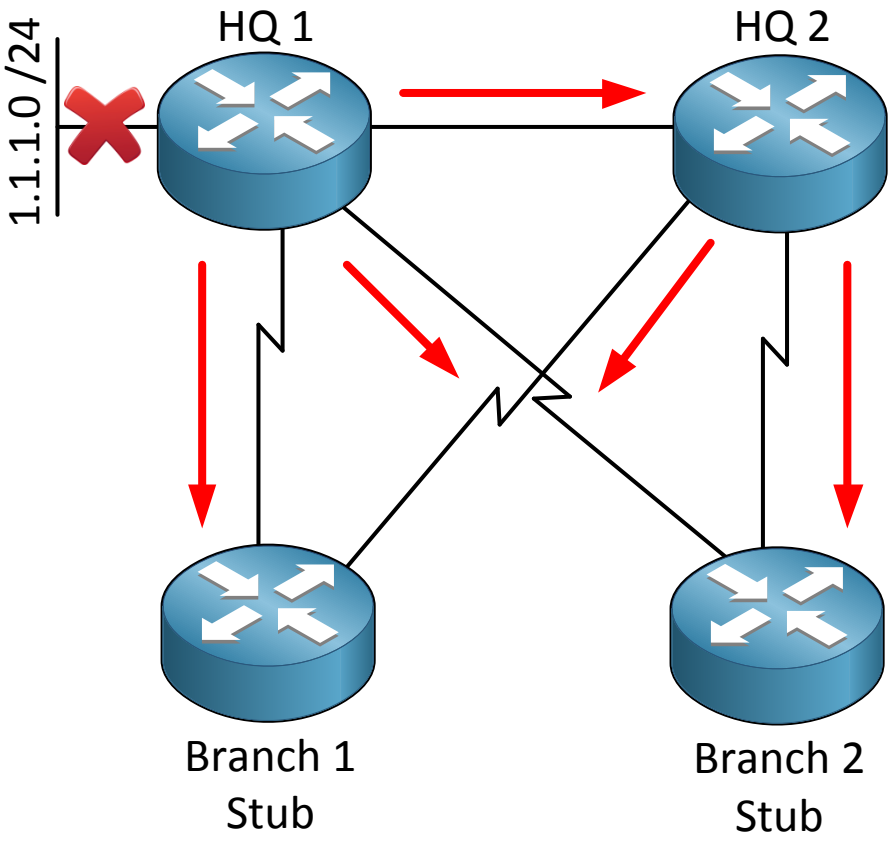
Now we have SIA query and SIA reply and things will work a little bit different:

1. Router 1 loses information about a network and has no feasible successors.
2. Router 1 sends a query to router 2.
3. Router 2 doesn't know the answer so sends a query as well to router 3.
4. Router 3 doesn't know the answer and sends a reply to router 2.
5. Router 2 sends a reply to router 1 to let him know he has no clue about this network.
6. Because of congestion the reply from router 2 never makes it back to router 1.
7. After 1.5 minute router 1 will send a SIA query to router 2 to ask for its status.
8. Router 2 will respond with a SIA reply and the neighbor adjacency will not be dropped.

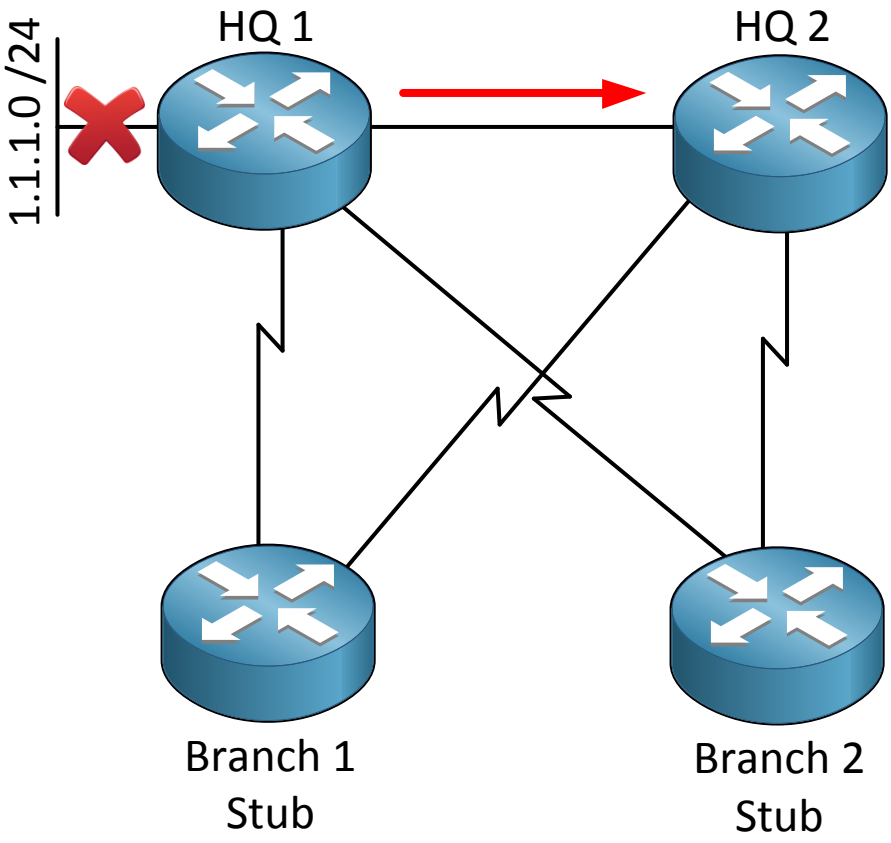
Does this make sense to you? There is something else we can do to stop queries….**EIGRP stub** to the rescue!



Look at the following topology. We have a company with a headquarters and 2 branch offices. The 2 HQ routers are connected on the LAN using a Gigabit link. The branch offices are connected to both HQ routers using slow 64kbps serial links. What do you think will happen once HQ 1 loses its route to the 1.1.1.0 /24 network on the left side?



Query packets (red arrows) will fly everywhere! It doesn't sound like a good idea to use the serial links for backup so we are going to change this behavior by turning the branche routers into **EIGRP stubs.**



If we configure the branch routers as stub routers they will **not receive queries** from our HQ routers. This is a very good technique to stop query traffic!

Why do we call it a stub? You have probably seen one before!

A tree that is cut off is called a stub…there's nothing attached to it anymore.

If you look at our branch routers they are like the tree. There are no other routers behind the branch routers and we don't want to use them as backup paths so why bother querying them?

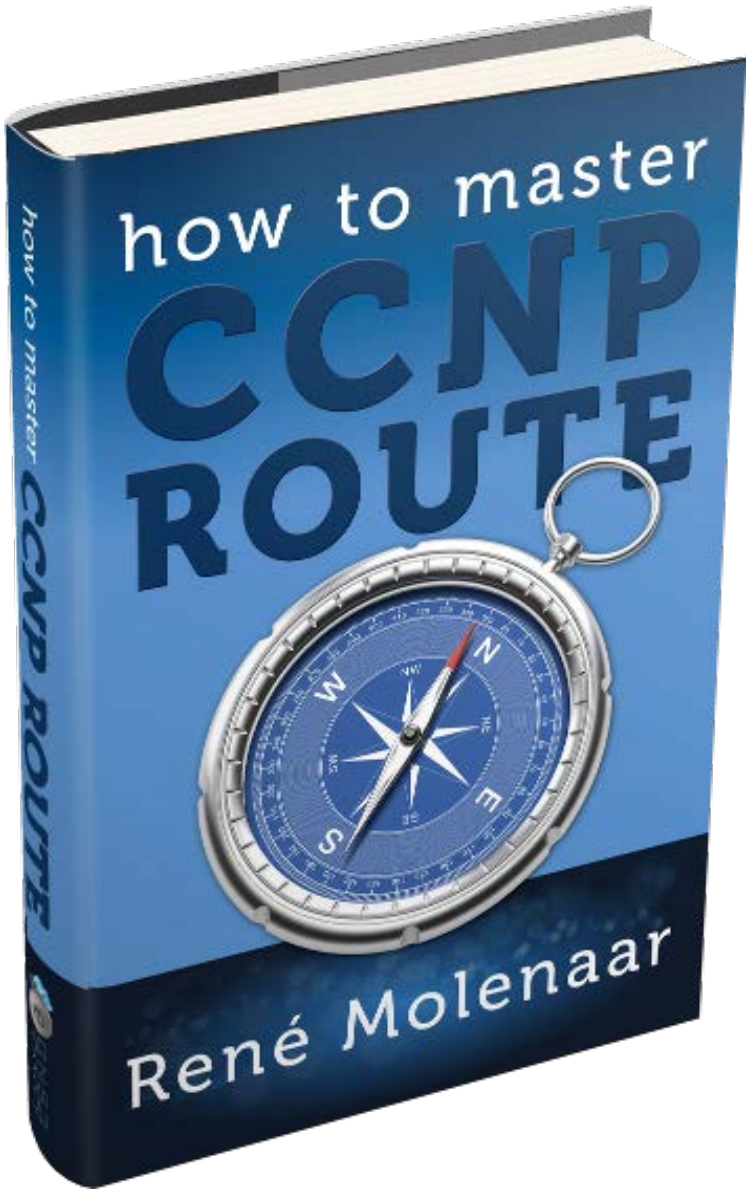
EIGRP stubs are not an "all or nothing" solution. We have different flavors so you can choose to which types of routes the stub router should receive queries or not.
Here are the flavors we have:

- **Receive-only**: The stub router will not advertise any network.
- **Connected**: allows the stub router to advertise directly connected networks.
- **Static**: allows the stub router to advertise static routes (you have to redistribute them).
- **Summary**: allows the stub router to advertise summary routes.
- **Redistribute**: allows the stub router to advertise redistributed routes.

Do you enjoy reading this sample of How to Master CCNP ROUTE ?

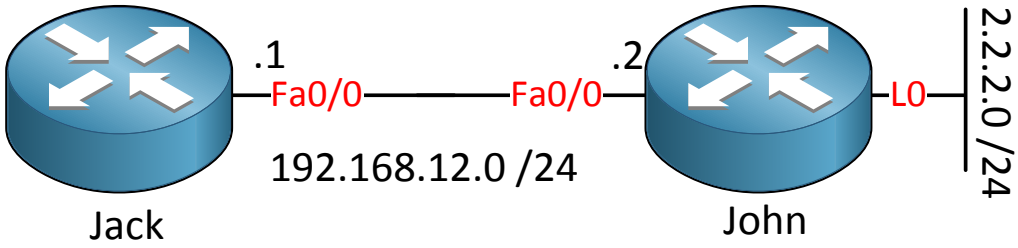
Click on the link below to get the full version.

Get How to Master CCNP ROUTE Today



The default is **connected** + **summary.** If you like you can mix some of the options with the exception of receive-only because it denies all advertisements. Redistribution is the importing and exporting of routing information from one routing protocol to another. This is something we will discuss later in the redistribution chapter.

Let's look at some EIGRP stub scenarios!




```
Jack(config)#router eigrp 123
Jack(config-router)#network 192.168.12.0
```

```
John(config-if)#router eigrp 12
John(config-router)#network 192.168.12.0
John(config-router)#network 2.2.2.0 0.0.0.255
```

I'll use the topology above and a basic EIGRP configuration. Let me first demonstrate the EIGRP query behavior to you.

```
Jack#debug eigrp packets query
EIGRP Packets debugging is on
    (QUERY)
```

```
John#debug eigrp packets query
EIGRP Packets debugging is on
    (QUERY)
```

First let me enable the debug on router Jack and John so you can see the queries.

```
John(config)#interface loopback 0
John(config-if)#shutdown
```

We'll shut the loopback0 interface on router John to see what will happen.

```
John#
EIGRP: Enqueueing QUERY on FastEthernet0/0 iidbQ un/rely 0/1 serno 33-33
EIGRP: Enqueueing QUERY on FastEthernet0/0 nbr 192.168.12.1 iidbQ un/rely
0/0 peerQ un/rely 0/0 serno 33-33
EIGRP: Sending QUERY on FastEthernet0/0
  AS 12, Flags 0x0, Seq 55/0 idbQ 0/0 iidbQ un/rely 0/0 serno 33-33
```

You can see that router John is sending a query towards router Jack.

```
Jack# EIGRP: Received QUERY on FastEthernet0/0 nbr 192.168.12.2
```

```
  AS 12, Flags 0x0, Seq 55/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
```

And we see that router Jack has received the query. So far so good, this is normal EIGRP behavior. You lose a network…you ask your neighbors if they know where it is.

```
John(config)#interface loopback 0
John(config-if)#no shutdown
```

First I'll restore the loopback0 interface…

```
Jack(config)#router eigrp 12
Jack(config-router)#eigrp stub
```

And we'll configure router Jack as an EIGRP stub router.

```
Jack#show running-config | begin router eigrp
router eigrp 12
 network 192.168.12.0
 no auto-summary
 eigrp stub connected summary
```

Just so you know, if you configure EIGRP stub without any parameters then it will use "connected" and "summary" by default.

```
Jack#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2 (FastEthernet0/0)
is down: peer info changed
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2 (FastEthernet0/0)
is up: new adjacency
John#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.1 (FastEthernet0/0)
is down: Interface Goodbye received
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.1 (FastEthernet0/0)
is up: new adjacency
```

You can see that configuring the stub feature breaks the EIGRP neighbor adjacency. Let's see if anything is different if I shut the loopback0 interface…

```
John(config)#interface loopback 0
John(config-if)#no shutdown
```

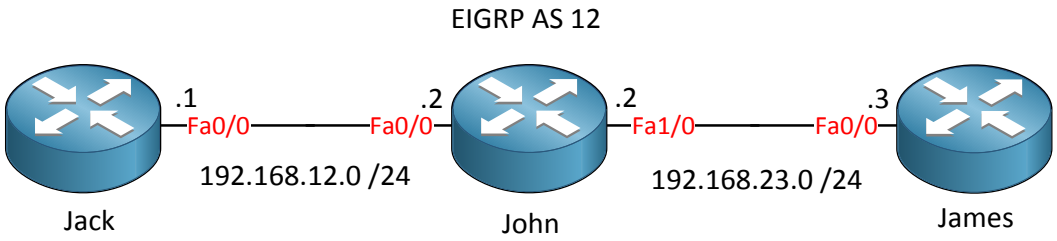
The interface goes down…let's see what will happen…

```
Jack# EIGRP: Enqueueing QUERY on FastEthernet0/0 iidbQ un/rely 0/1 serno
28-28
EIGRP: Enqueueing QUERY on FastEthernet0/0 nbr 192.168.12.2 iidbQ un/rely
0/0 peerQ un/rely 0/0 serno 28-28
EIGRP: Sending QUERY on FastEthernet0/0
  AS 12, Flags 0x0, Seq 44/0 idbQ 0/0 iidbQ un/rely 0/0 serno 28-28
```

```
John#
EIGRP: Received QUERY on FastEthernet0/0 nbr 192.168.12.1
  AS 12, Flags 0x0, Seq 44/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
```

You can see that router Jack will send a query towards router John because it has lost the 2.2.2.0 /24 network. It's not receiving a query anymore from router John because it's a stub router!

Now you know how it deals with the queries, let's take a look at the different stub options!



I'll use a different topology to demonstrate the different stub options. Here are the EIGRP configurations of the 3 routers:

```
Jack#show run | begin router eigrp
router eigrp 12
 network 192.168.12.0
 no auto-summary
John#show run | begin router eigrp
router eigrp 12
 network 2.2.2.0 0.0.0.255
 network 192.168.12.0
 no auto-summary
```

```
James#show run | begin router eigrp
router eigrp 12
 network 3.3.3.0 0.0.0.255
 network 192.168.23.0
 no auto-summary
```

Nothing special as you can see. Let me add a loopback0 interface on router Jack and advertise it into EIGRP.

```
Jack(config)#interface loopback 0
Jack(config-if)#ip address 1.1.1.1 255.255.255.0
```

```
Jack(config)#router eigrp 12
Jack(config-router)#network 1.1.1.0 0.0.0.255
```

Nothing special so far…

```
John#show ip route eigrp
     1.0.0.0/24 is subnetted, 1 subnets
D       1.1.1.0 [90/156160] via 192.168.12.1, 00:00:48, FastEthernet0/0
```

You can see that router John has learned about network 1.1.1.0 /24. Now let's play with the stub feature.

```
Jack(config)#router eigrp 12
Jack(config-router)#eigrp stub receive-only
```

Let's make it receive-only.

```
John#show ip route eigrp

```

Easy enough…router Jack doesn't advertise anything anymore so router John doesn't know about the 1.1.1.0 /24 network anymore. That's it…let's try the next stub option:

```
Jack(config)#router eigrp 12
Jack(config-router)#no eigrp stub receive-only
```

Let's start by getting rid of the stub feature on router Jack.

```
James#show ip route eigrp
D    192.168.12.0/24 [90/30720] via 192.168.23.2, 00:12:49, FastEthernet0/0
     1.0.0.0/24 is subnetted, 1 subnets
D       1.1.1.0 [90/158720] via 192.168.23.2, 00:11:16, FastEthernet0/0
```

Let's take a quick look at the routing table of router James. It has learned two networks through EIGRP:

- 192.168.12.0 /24 which is the link between router Jack and John.
- 1.1.1.0 /24 which is the loopback0 interface of router Jack.

```
John(config)#router eigrp 12
John(config-router)#eigrp stub connected
```

Let's enable the eigrp stub connected option on router John.

```
John#show ip route eigrp
     1.0.0.0/24 is subnetted, 1 subnets
D       1.1.1.0 [90/156160] via 192.168.12.1, 00:10:45, FastEthernet0/0
```

```
James#show ip route eigrp
D    192.168.12.0/24 [90/30720] via 192.168.23.2, 00:01:03, FastEthernet0/0
```

Interesting…router James only has 192.168.12.0 /24 in its routing table now. Why? It's because this network is directly connected to router John and that's why it's advertised. 1.1.1.0/24 is not advertised from router John to James because of the stub connected option!

```
John(config)#router eigrp 12
John(config-router)#no eigrp stub connected
```

Let's clean up before we continue with the next stub option…

```
John(config)#router eigrp 12
John(config-router)#eigrp stub static
```

This time I'll turn router John into a stub static. Let's see what router James thinks of this…

```
James#show ip route eigrp
```

Router James doesn't have any EIGRP information anymore. Why? It's because router John only advertises redistributed static routes…

```
John#show ip route eigrp
     1.0.0.0/24 is subnetted, 1 subnets
D       1.1.1.0 [90/156160] via 192.168.12.1, 00:01:29, FastEthernet0/0
```

You can see that router John still knows about network 1.1.1.0/24…

```
John(config)#ip route 1.1.1.0 255.255.255.0 192.168.12.1
```

First I'll create a static route and point it towards router Jack.

```
John(config)#router eigrp 12
John(config-router)#redistribute static
```

Secondly we'll make sure that router John redistributes this static route into EIGRP.

```
James#show ip route eigrp
     1.0.0.0/24 is subnetted, 1 subnets
D EX    1.1.1.0 [170/30720] via 192.168.23.2, 00:01:06, FastEthernet0/0
```

There we go…since router John only advertises redistributed static routes we can now see network 1.1.1.0/24 again in the routing table of router James.

```
John(config)#router eigrp 12
John(config-router)#no eigrp stub static
John(config-router)#no redistribute static
John(config-router)#exit
John(config)#no ip route 1.1.1.0 255.255.255.0 192.168.12.1
```

Let's clean up before we continue.

```
John(config)#router eigrp 12e.
John(config-router)#eigrp stub summary
```

Let's turn router John into an EIGRP stub summary router.

```
James#show ip route eigrp
```

Of course router James is clueless again because router John will only advertise summary routes…

```
John(config)#interface fastEthernet 1/0
John(config-if)#ip summary-address eigrp 12 1.1.0.0 255.255.0.0
```

We'll configure a summary and advertise it towards router James.

```
James#show ip route eigrp
     1.0.0.0/16 is subnetted, 1 subnets
D       1.1.0.0 [90/158720] via 192.168.23.2, 00:01:10, FastEthernet0/0
```

You can see the summary in the routing table of router James…that's it!

```
John(config)#interface fastEthernet 1/0
John(config-if)#no ip summary-address eigrp 12 1.1.0.0 255.255.0.0
John(config-if)#exit
John(config)#router eigrp 12
John(config-router)#no eigrp stub summary
```

Let's clean up and move on to the final EIGRP stub option, redistribute.

```
John(config)#router eigrp 12
John(config-router)#eigrp stub redistributed
```

Let's enable the redistributed stub option.

```
John#show run | include eigrp stub
 eigrp stub connected summary redistributed
```

First thing you might notice is that it also includes connected and summary routes.

Let's add a loopback interface on router John and redistribute it into EIGRP:

```
John(config)#interface loopback 1
John(config-if)#ip address 2.2.2.2 255.255.255.0
```

```
John(config)#router eigrp 12
John(config-router)#redistribute connected
```

Instead of the network command we'll redistribute the loopback interface into EIGRP.

```
James#show ip route | include 2.2.2.0
D EX    2.2.2.0 [170/156160] via 192.168.23.2, 00:00:42, FastEthernet0/0
```

You can see that router James learned the 2.2.2.0/24 network.

Are you following me so far? EIGRP stubs are quite fun to play with. If you want to see all the stubs in action you should try some of my labs or watch the videos I created:

http://gns3vault.com/EIGRP/eigrp-stub.html

http://gns3vault.com/EIGRP/eigrp-stub-leak-map.html

Phew! We made it to the end of the EIGRP chapter! What do you think? There's quite some material about EIGRP you have to understand if you want to master your CCNP ROUTE exam. Cisco exams are very practical-minded so make sure you practice on labs until you fully understand EIGRP. Throughout the chapters I gave you some links to EIGRP labs that

cover the different topics and now is a good time to try some of the full CCNP EIGRP labs I have for you.

These labs are a mix of all the different EIGRP items you have seen like summaries, authentication, stubs, load balancing and more. If you can beat those without too much effort you'll have mastered EIGRP!

http://gns3vault.com/CCNP/eigrp-ccnp-1.html

Here you can find all the EIGRP labs I currently have:

http://gns3vault.com/Table/EIGRP/

If you feel confident on EIGRP and are up for a challenge you should try my EIGRP troubleshooting lab:

http://gns3vault.com/Troubleshooting/eigrp-troubleshooting.html

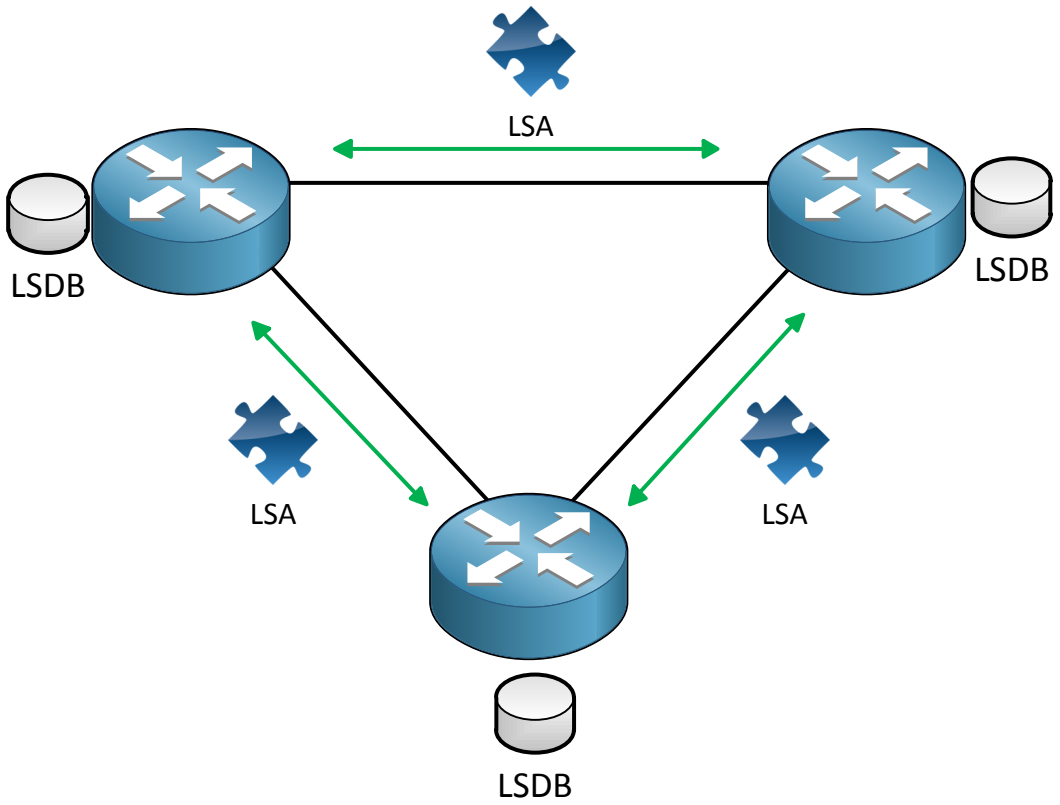
7. Introduction to OSPF

Next to EIGRP there is one more IGP (Interior Gateway Protocol) we are going to look at which happens to be **OSPF (Open Shortest Path First).** This first chapter will be an introduction. Some of the things you might have seen from CCNA but I've added some extras.

I don't know about you but I love my navigation system. The good thing about them is you can just drive and there is no need to look for traffic signs, the bad thing is that I'm absolutely lost when it's not working.

I'm bad at reading maps (or maybe I just don't like it) and if I had to find my way to some street in any big city I'm doomed.

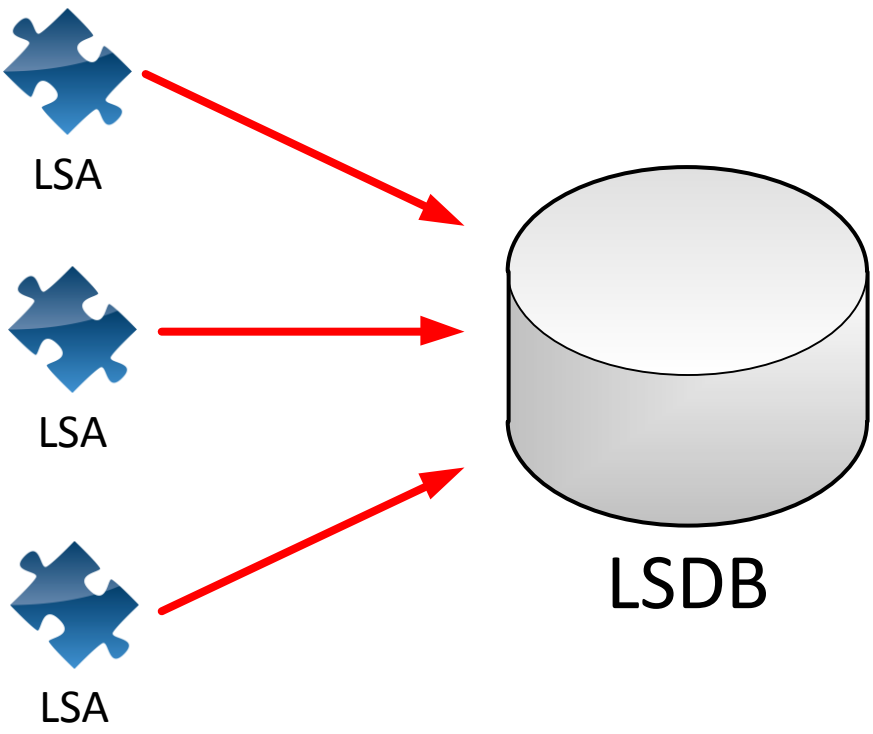
Link-state routing protocols are like your navigation system, they have a complete map of the network. If you have a full map of the network you can just calculate the shortest path to all the different destinations out there. This is cool because if you know about all the different paths it's impossible to get a loop since you know everything! The downside is that this is more CPU intensive than a distance vector routing protocol. It's just like your navigation system…if you calculate a route from New York to Los Angeles it's going to take a bit longer than when you calculate a route from one street to another street in the same city.



Let's take a good look at link-state and what it exactly means:

- ✓ Link: That's the interface of our router.
- ✓ State: Description of the interface and how it's connected to neighbor routers.

Link-state routing protocols operate by sending **link-state advertisements** (**LSA**) to all other link-state routers.

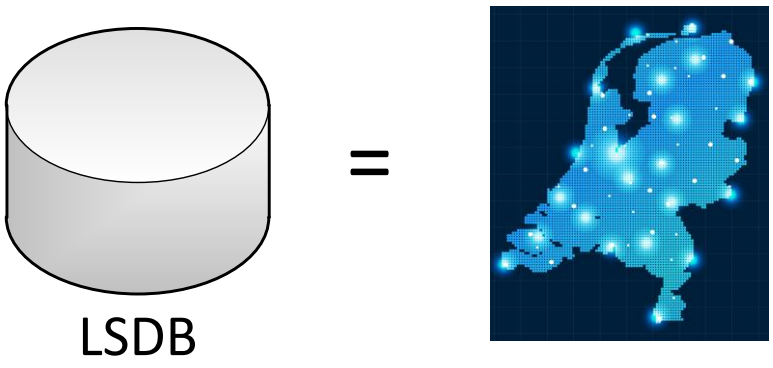


All the routers need to have these link-state advertisements so they can build their **link-state database** or **LSDB**.

Basically all the link-state advertisements are a piece of the puzzle which builds the LSDB.

This LSDB is our full picture of the network, in network terms we call this the **topology**.

You could compare the LSDB to having a full map of your country.

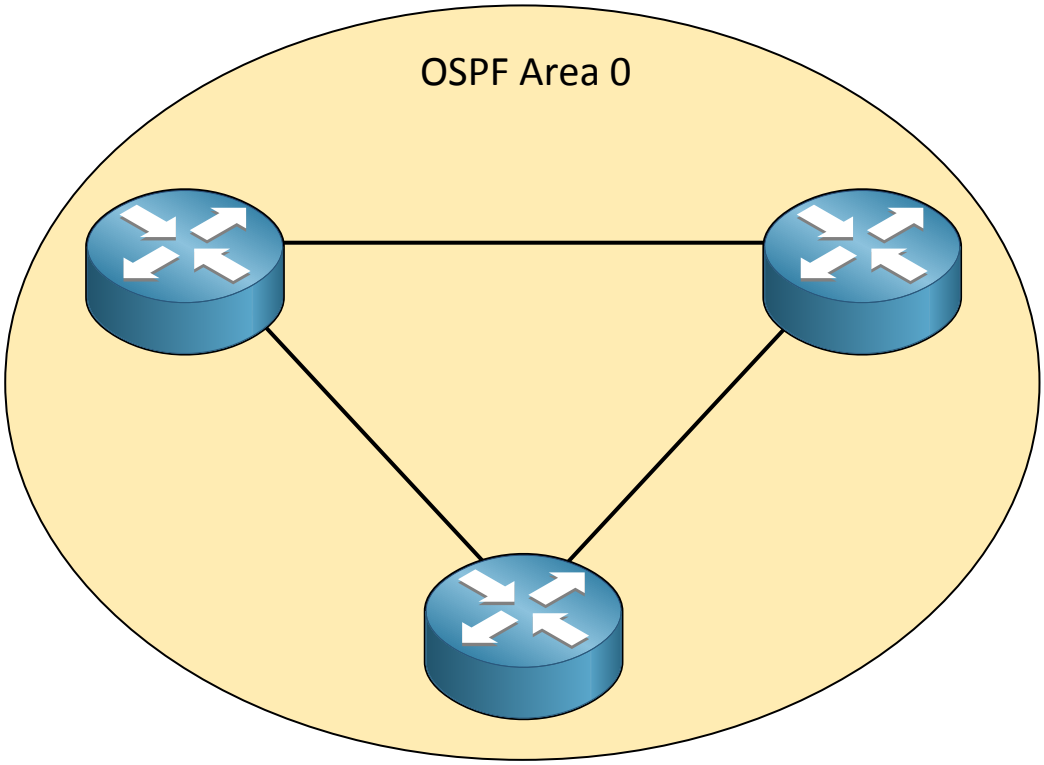


Once every router has a complete map we can start calculating the shortest path to all the different destinations by using the **shortest-path first (SPF) algorithm**. The BEST information goes into the routing table. Calculating the shortest path is like using your navigation system, it will look at the map and look at all the different ways of getting to the destination and only show you the best way of getting there.

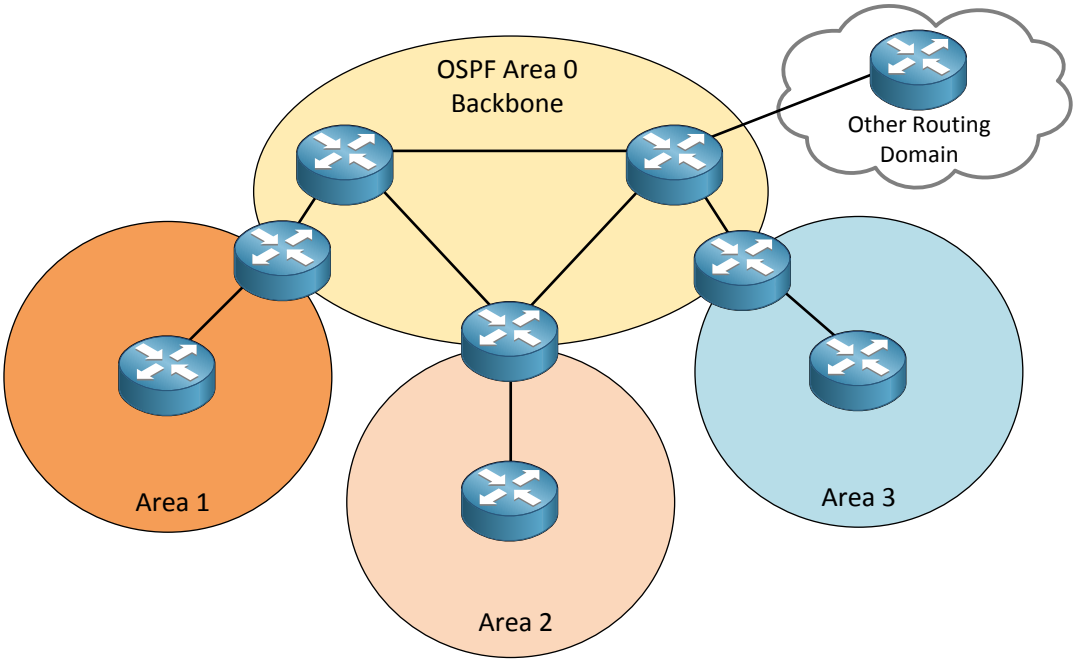
There is only one link-state routing protocol we are going to discuss which is OSPF (Open

Shortest Path First). There is another link-state routing protocol called IS-IS but it has been completely removed from CCNA, CCNP and even CCIE by Cisco.

Enough of my link-state routing protocol introduction let's take a look at OSPF and see how it operates.



OSPF works with the concepts of **areas** and by default you will always have a single area, normally this is area 0 or also called the **backbone** area.



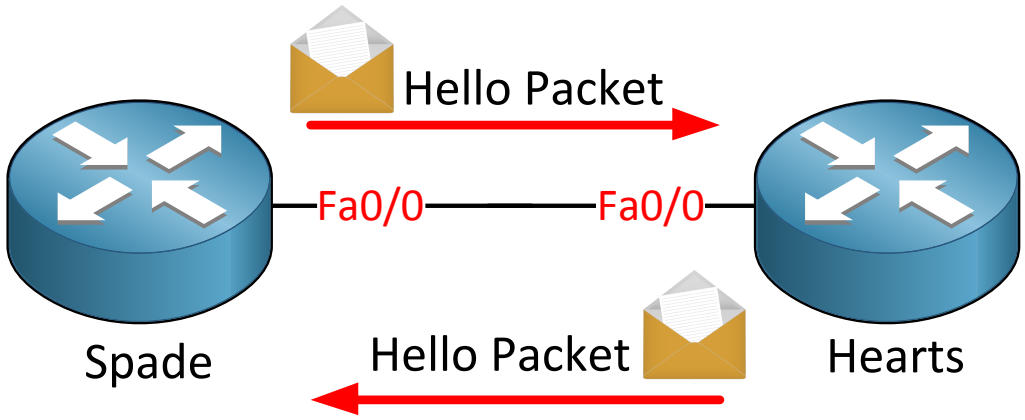
You can have multiple areas however as in the picture above, we have area 1, 2 and 3. All of these areas **must connect** to the backbone area. If you want to go from area 1 to area 2 you **must** go through the backbone area to get there. It's impossible to go from area 1 directly to area 2; you always have to pass the backbone area! Same thing if you want to go from area 3 to area 2…you must cross the backbone area.

So why do we work with areas? Remember what I just explained about your navigation system. If you tell your navigation system to calculate from New York to Los Angeles it will take much longer than calculating a route from one street to another street in the same city. This calculating is called the **shortest path first** or **SPF** algorithm and the same thing apply to OSPF. Our routers only have a full picture of the network topology within the area, the smaller your map the faster your SPF algorithm works!

Keep in mind the SPF algorithm is from the 70's and OSPF was invented somewhere in the 80's…we didn't have fancy Core 2 Duo / Quad and I7's back then.

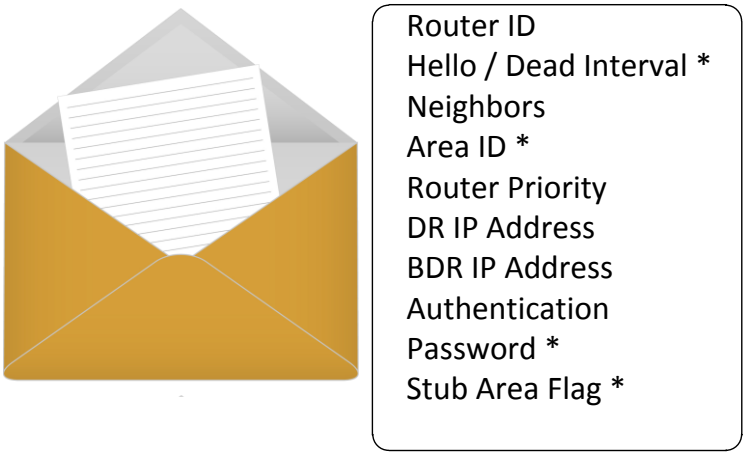
In the picture you also see on the top right something called "other routing domain". This could be another network running another routing protocol (perhaps RIP) and it's possible to import and export routes from RIP into OSPF or the other way around, this is called **redistribution.**

- ✓ Routers in the backbone area (area 0) are called backbone routers.
- ✓ Routers between two areas (like the one between area 0 and area 1) are called **area border routers** or **ABR.**
- ✓ Routers that run OSPF and are connected to another network that runs another routing protocol (for example RIP) are called **autonomous system border routers** or **ASBR.**



OSPF works differently than RIP or EIGRP, first of all it's a link-state compared to a distance vector but it also doesn't just send the link-state-advertisements around. Routers have to be become neighbors first; once we have become neighbors we are going to exchange link-state advertisements.

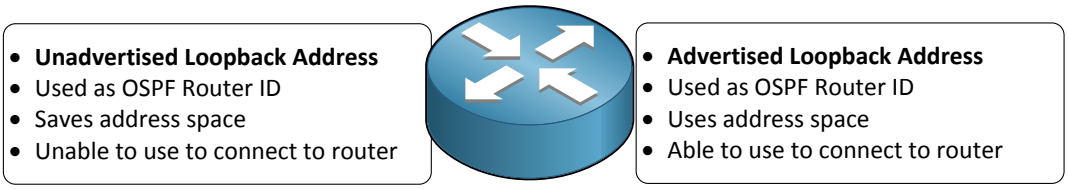
Once you configure OSPF your router will start sending hello packets. If you also receive hello packets from the other router you will become neighbors.



There's one catch however, there are a couple of fields in the hello packet and many of them have to match otherwise you won't become neighbors. Let's walk through the items in the hello packet and see what they are about:

- ✓ **Router ID:** Each OSPF router needs to have an unique ID which is the highest IP address on any active interface. More about this later.
- ✓ **Hello / Dead Interval:** Every X seconds we are going to send a hello packet, if we don't hear any hello packets from our network for X seconds we declare you "dead" and we are no longer neighbors. These values have to match on both sides in order to become neighbors.
- ✓ **Neighbors**: All other routers who are your neighbors are specified in the hello packet.
- ✓ **Area ID**: This is the area you are in. This value has to match on both sides in order to become neighbors.

- ✓ **Router Priority**: This value is used to determine who will become designated or backup designated router. More on this later.
- ✓ **DR and BDR IP address**: Designated and Backup Designated router. More on this later.
- ✓ **Authentication password:** You can use clear text and MD5 authentication for OSPF which means every packet will be authenticated. Obviously you need the same password on both routers in order to make things work.
- ✓ **Stub area flag:** Besides area numbers OSPF has different area types, we will cover this later. Both routers have to agree on the area type in order to become neighbors.



Each OSPF router needs to have a unique router ID which is based on the **highest IP address** on any **active interface**.

If you have a loopback interface on your OSPF router then this IP address will be used as the router ID even when it's not the highest active IP address. Why does OSPF do this? Well it makes sense…your loopback interface will never go down unless your router crashes.

Using a loopback interface you can do 2 things:

- ✓ Advertise the IP address on the loopback interface in OSPF.
- ✓ Don't advertise the IP address on the loopback interface in OSPF.

What's the difference? Well if you advertise it other routers will be able to reach and ping the IP address on this loopback interface or even use it to telnet into the router. If you don't then well you can't…it's as easy as that.

Everything is well, we have configured OSPF…we have become neighbors with a bunch of routers and they have exchanged link-state advertisements. Our routers have built their LSDB and they have a full topology picture of our network. Next step is to run the SPF algorithm and see what the shortest path to our destination is.
What about the metric? OSPF uses a metric called **cost** which is based on the bandwidth of an interface, it works like this:

Cost = Reference Bandwidth / Interface Bandwidth

The reference bandwidth is a default value on Cisco routers which is a 100Mbit interface. You divide the reference bandwidth by the bandwidth of the interface and you'll get the cost.

Example: If you have a 100 Mbit interface what will the cost be?

Cost = Reference bandwidth / Interface bandwidth.

100 Mbit / 100 Mbit = COST 1

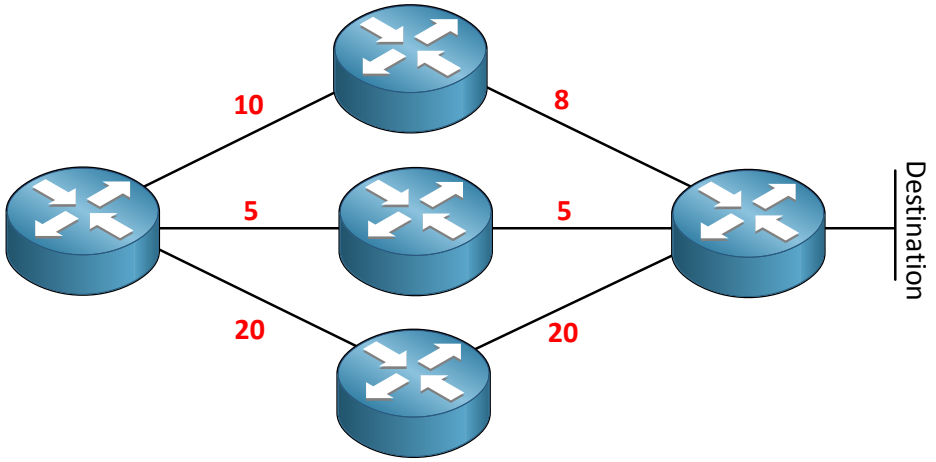
Example: If you have a 10 Mbit interface what will the cost be?

100 Mbit / 10 Mbit = COST 10

Example: If you have a 1 Mbit interface what will the cost be?

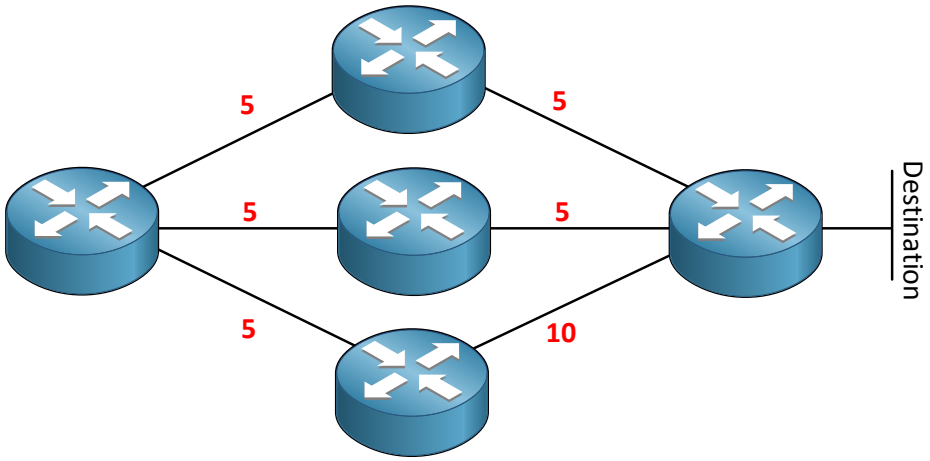
100 Mbit / 1 Mbit = COST 100

The **lower** the cost the better the path is.



Look at the picture above, we are sitting in the router on the left and running the SPF algorithm looking for the shortest path to our destination. Which one are we going to use?

Using the router on top we would have a cost of 10+8 which is 18. The path in the middle is 5+5 = 10. The router on the bottom we would have a cost of 20+20 = 40. The path in the middle obviously has the lowest cost so this is the path we are going to use!



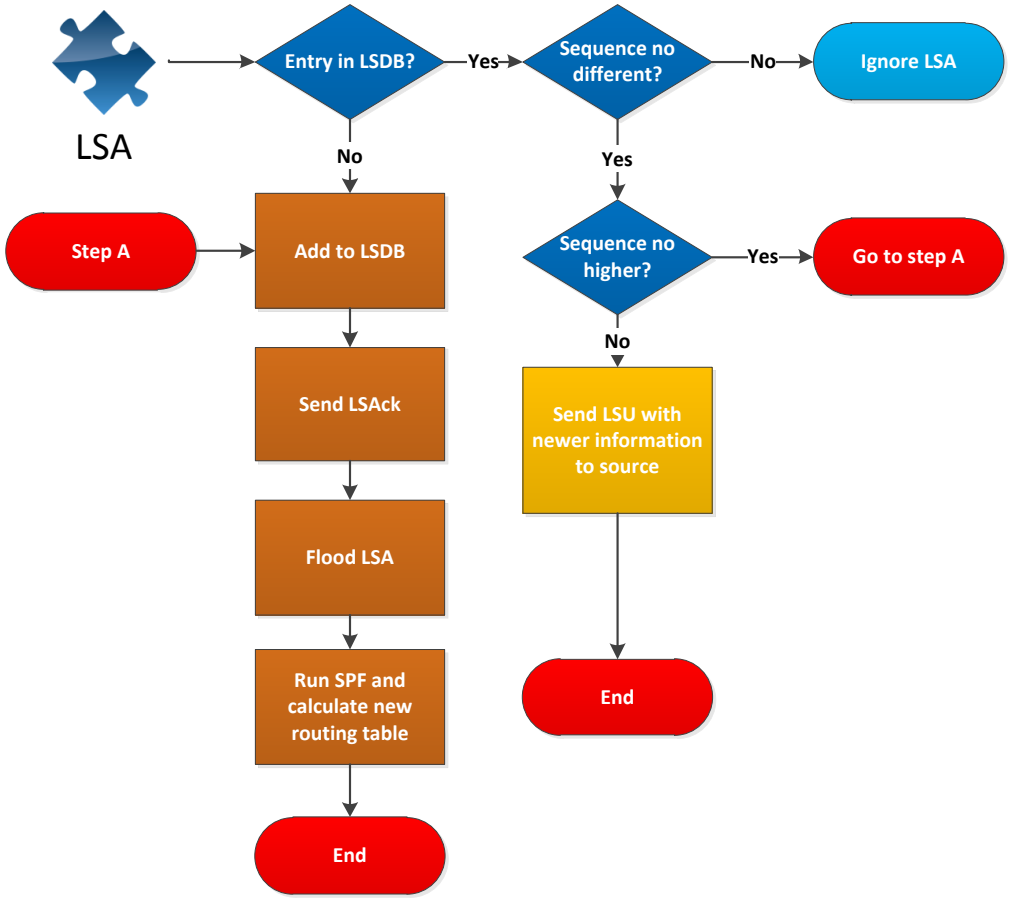
Look at this picture: As you can see the path through the router on top and the middle router have the same cost (5+5 = 10). What are we going to do here? The path is equal.

The answer is **load balancing**! We are going to use both paths and OSPF will load balance among them 50/50. Some things worth knowing about OSPF load balancing:

- ✓ Paths must have an equal cost.
- ✓ 4 equal cost paths will be placed in routing table.
- ✓ Maximum of 16 paths.
- ✓ To make paths equal cost, change the "cost" of a link

If a path is not equal we can make it so by manually changing the cost or bandwidth of an interface.

How exactly does OSPF fill the LSDB? Let's zoom in on the operation of how OSPF keeps its link-state database up-to-date.



Each LSA has an **aging timer** which carries the **link-state age field.** By default each OSPF LSA is only valid for **30 minutes.** If the LSA expires then the router that created the LSA will resend the LSA and increase the **sequence number**.

Let's walk through this flowchart together. In this example a new LSA is arriving at the router and OSPF has to decide what to do with it:

1. If the LSA isn't already in the LSDB it will be added and a LSAck (acknowledgement) will be sent to the OSPF neighbor. The LSA will be flooded to all other OSPF neighbors and we have to run SPF to update our routing table.
2. If the LSA is already in the LSDB and the sequence number is the same then we will ignore the LSA.
3. If the LSA is already in the LSDB and the sequence number is different then we have to take action:
    a. If the sequence number is higher it means this information is newer and we have to add it to our LSDB.
    b. If the sequence number is lower it means our OSPF neighbor has an old LSA and we should help them. We will send a **LSU** (**Link state update)** including the newer LSA to our OSPF neighbor. The LSU is an envelope that can carry multiple LSAs in it. More about OSPF packets in the next chapter!

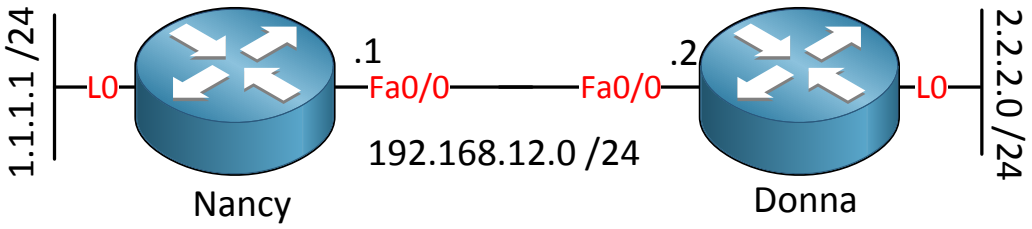
It's not just the sequence number that OSPF will look at to determine if a LSA is **more recent**. It will consider the LSA to be more recent if it has:

- A higher sequence number.
- A higher checksum number.
- An age equal to the maximum age.
- If the link-state age is much younger.

What do the sequence numbers look like for OSPF LSAs?

- There are 4 bytes or 32-bits.
- Begins with 0x80000001 and ends at 0x7FFFFFFF.
- Every 30 minutes each LSA will age out and will be flooded:
    - The sequence number will increment by one.

With 32-bits we have a LOT of sequence numbers and every 30 minutes it will increase. If we make it to the last sequence number 0x7FFFFFFF it will wrap around and start again at 0x80000001. Every 30 minutes OSPF will flood a LSA to make sure the LSDB stays up to date and when it does this the sequence number will increase and OSPF will reset the max age when it receives a new LSA update.



Let's use the topology above to configure OSPF and see some LSAs in action.

```
Nancy(config)#router ospf 1
Nancy(config-router)#network 0.0.0.0 255.255.255.255 area 0
Nancy(config-router)#exit
```

```
Donna(config)#router ospf 1
Donna(config-router)#network 0.0.0.0 255.255.255.255 area 0
Donna(config-router)#exit
```

I'm using the above network command to advertise any interface with an IP address, keep in mind this is not best practice. It's just my "quick and dirty" method to get OSPF going…I like to do this if I only care about "connectivity".

```
Donna#show ip ospf database

            OSPF Router with ID (2.2.2.2) (Process ID 1)

           Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
1.1.1.1         1.1.1.1         109         0x80000002 0x004ED8 2
2.2.2.2         2.2.2.2         108         0x80000002 0x003EDB 2

           Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
192.168.12.2    2.2.2.2         108         0x80000001 0x008F1F
```

Here is the output of the **show ip ospf database** command which shows us the LSDB. What interesting things can we see here?

- Each OSPF router has interfaces in an area. For each interface it will advertise a **router LSA**. The link ID is the OSPF router ID. Since I'm using loopback interfaces on both routers that IP address was chosen for the OSPF router ID.
- The first router LSA entry you see is from router Nancy and you can see the age, sequence number and a checksum. You can see that the LSA has been updated 2 times since the sequence number is 0x80000002 (I did a **clear ip ospf process** to demonstrate this).

So far my OSPF introduction... Throughout the OSPF chapters I'm going to tell a bit more about the OSPF database. In the upcoming chapter we'll start with looking at the different OSPF packets.

If you want to look at the OSPF LSDB yourself and play a bit you can use one of my OSPF single area labs:

http://gns3vault.com/OSPF/ospf-single-area.html

If you need more refreshment you can try one of the full OSPF CCNA labs:

http://gns3vault.com/CCNA/ospf-for-ccna-1.html

8. OSPF Packets and Neighbor discovery

In this chapter I'm going to show you the different packets OSPF uses and how neighbor discovery works.

OSPF uses its own protocol like EIGRP and doesn't use a transport protocol like TCP or UDP. If you would look at the IP packet in wireshark you can see that OSPF has **protocol ID 89** for all its packets.

```
Donna#debug ip ospf packet
OSPF packet debugging is on
      OSPF: rcv. v:2 t:1 l:48 rid:1.1.1.1
      aid:0.0.0.0 chk:4D40 aut:0 auk: from FastEthernet0/0
```

If we use **debug ip ospf packet** we can look at the OSPF packet on our router. Let's look at the different fields we have:

- **V:2** stands for OSPF version 2. If you are running IPv6 you'll version 3.
- **T:1** stands for OSPF packet number 1 which is a hello packet. I'm going to show you the different packets in a bit.
- **L:48** is the packet length in bytes. This hello packet seems to be 48 bytes.
- **RID 1.1.1.1** is the Router ID.
- **AID** is the area ID in dotted decimal. You can write the area in decimal (area 0) or dotted decimal (area 0.0.0.0).
- **CHK 4D40** is the checksum of this OSPF packet so we can check if the packet is corrupt or not.
- **AUT:0** is the authentication type. You have 3 options:
  - 0 = no authentication
  - 1 = clear text
  - 2 = MD5
- **AUK:** If you enable authentication you'll see some information here.

Let's continue by looking at the different OSPF packet types:

1. Hello
2. Database Description (DBD)
3. Link-State Request (LSR)
4. Link-State Update (LSU)
5. Link-State Acknowledgment (LSAck)

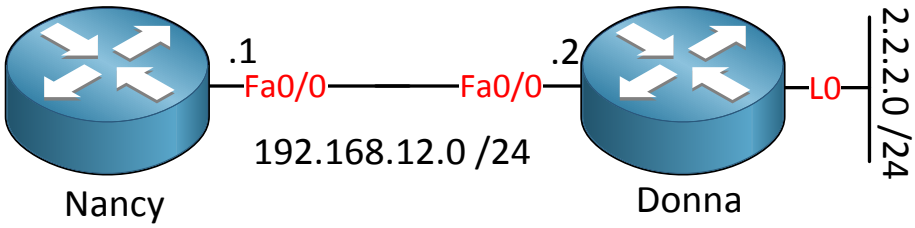
I'm throwing them right at you; here are all the **OSPF packet types** we have. In my **debug ip ospf packet** at the previous page you could see T:1 which stands for packet type 1. Here you see that it corresponds to an OSPF hello packet. What is the role of each OSPF packet?

- **Hello:** neighbor discovery, build neighbor adjacencies and maintain them.
- **DBD:** This packet is used to check if the LSDB between 2 routers is the same. The DBD is a **summary of the LSDB**.
- **LSR:** Requests specific link-state records from an OSPF neighbor.
- **LSU:** Sends specific link-state records that were requested. This packet is like an envelope with multiple LSAs in it.
- **LSAck:** OSPF is a reliable protocol so we have a packet to acknowledge the others.

OSPF has to get through 7 states in order to become neighbors…here they are:

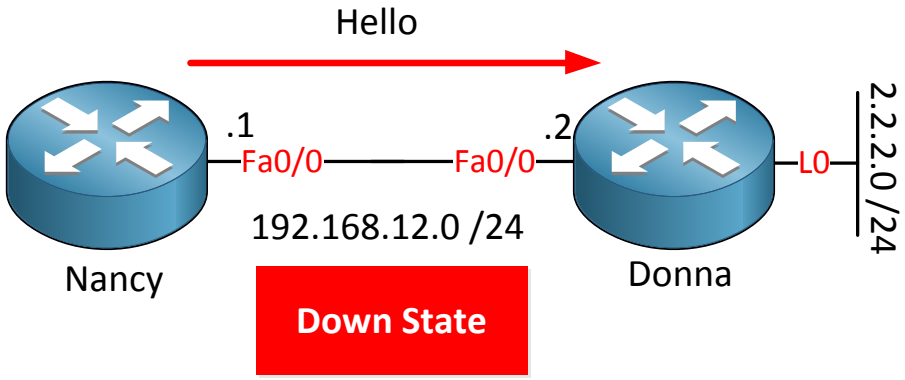
1. **Down**: no OSPF neighbors detected at this moment.
2. **Init:** Hello packet received.
3. **Two-way:** own router ID found in received hello packet.
4. **Exstart:** master and slave roles determined.
5. **Exchange:** database description packets (DBD) are sent.
6. **Loading:** exchange of LSRs (Link state request) and LSUs (Link state update) packets.
7. **Full:** OSPF routers now have an adjacency.

Let's have a detailed look at this process!

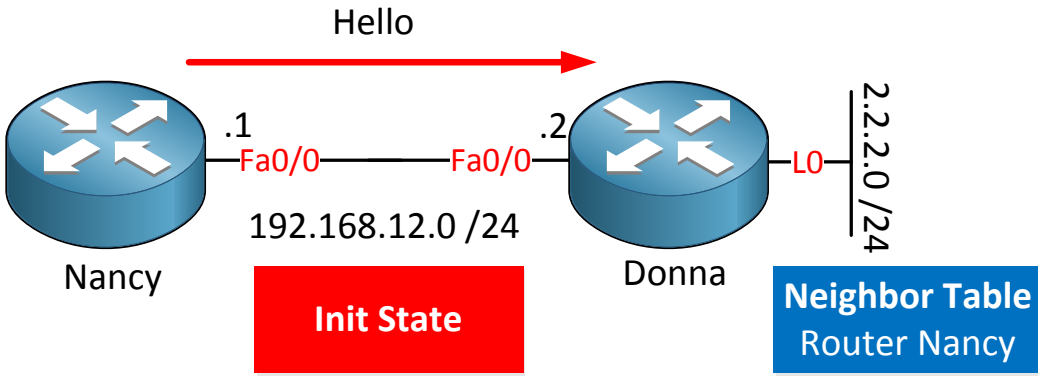


This is the topology I'm using. Router Nancy and Donna are connected using a single link

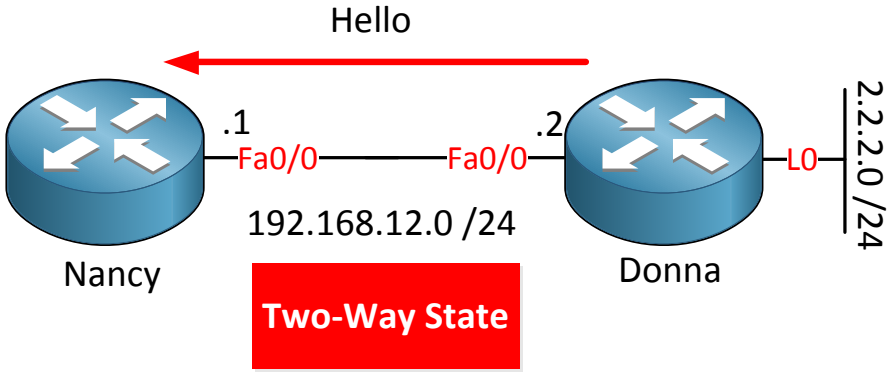
and we will see how router Nancy learns about the 2.2.2.0 /24 network.



As soon as I configure OSPF on router Nancy it will start sending hello packets. Router Nancy has no clue about other OSPF routers at this moment so it's in the **down state**. The hello packet will be sent to the **multicast address 224.0.0.5.**



Router Donna receives the hello packet and will put an entry for router Nancy in the **OSPF neighbor table.** We are now in the **init state.**

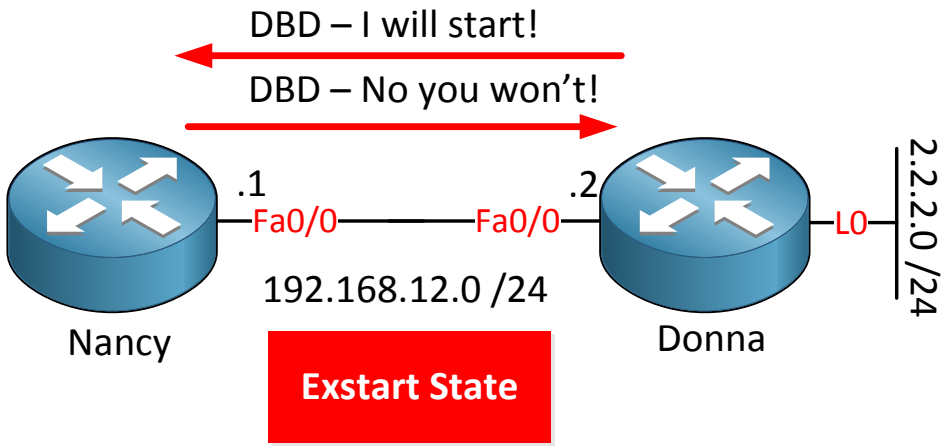


Router Donna has to respond to router Nancy with a hello packet. This packet is not sent using multicast but with **unicast** and in the neighbor field it will include **all OSPF neighbors** that router Donna has. Router Nancy will see **her own name** in the neighbor field in this hello packet.

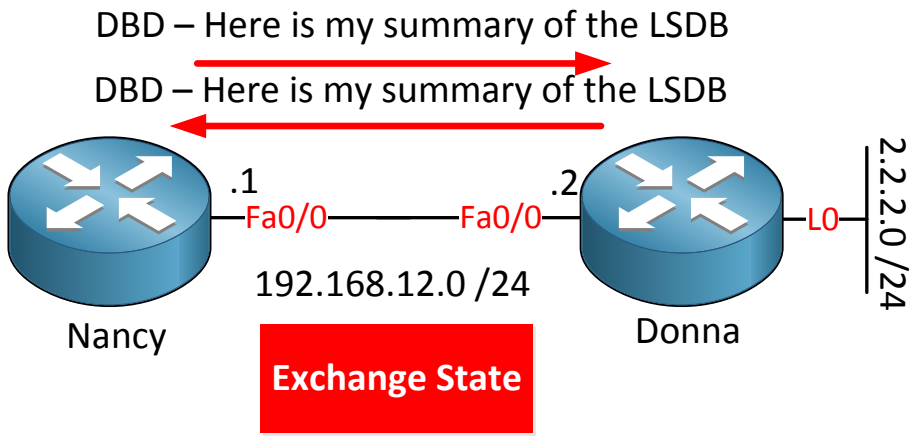
Router Nancy will receive this hello packet and sees her own router ID. We are now in the

two-way state.

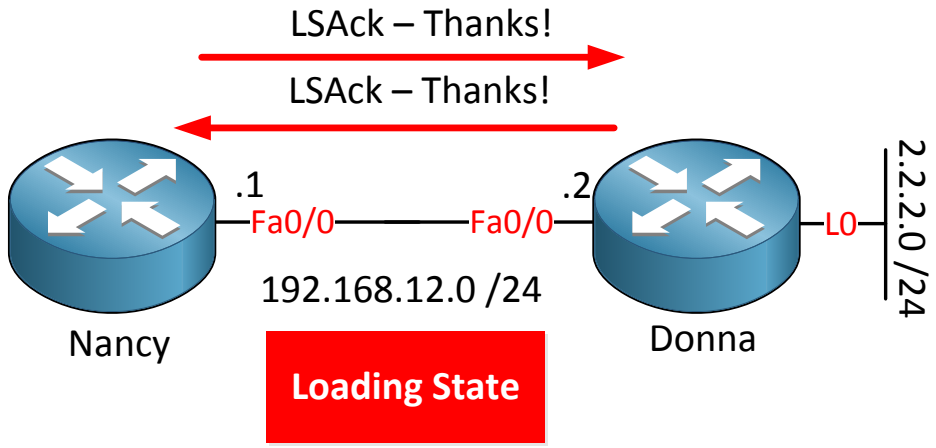
I have to take a pause here. If the link we are using is a **multi-access network** OSPF has to elect a **DR (Designated Router)** and **BDR (Backup Designated Router)**. This has to happen before we can continue with the rest of the process. In the next chapter I'm going to teach your DR/BDR…for now just hold the thought that the DR/BDR election happens right after the two-way state ok?



Our next stop is the **exstart state**. Our routers are ready to sync their LSDB. At this step we have to select a **master** and **slave** role. The router with the highest router ID will become the master. Router Donna has the highest router ID and will become the master.

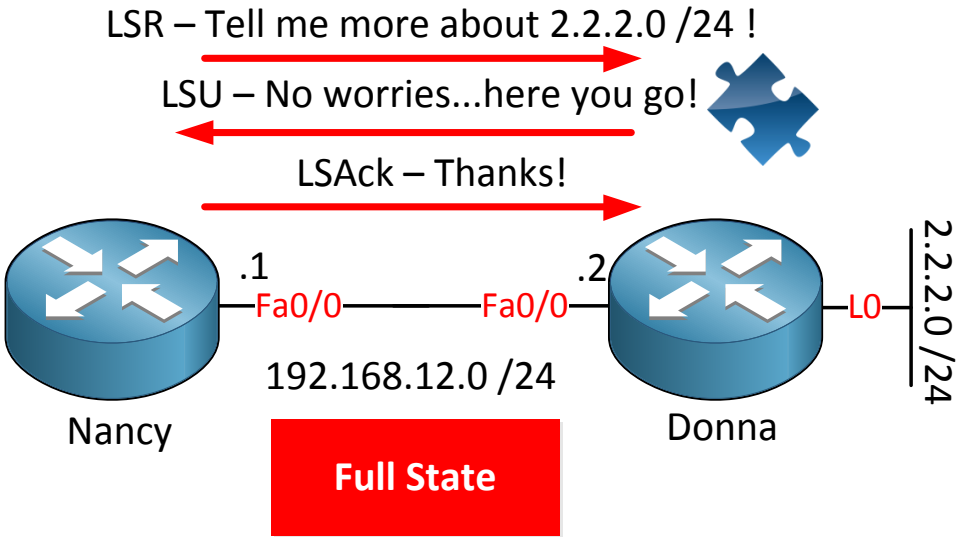


In the **exchange state** our routers are sending a DBD with a summary of the LSDB. This way the routers can find out what networks they don't know about.



When our routers receive the DBD from the other side they will do a couple of things:

- Send an acknowledgement using the LSAck packet.
- Compare the information in the DBD with the information it already has:
  - If the neighbor has new or newer information it will send a LSR (Link State Request) packet to request for this information
- When the routers start sending a LSR (Link State Request) we are in the **loading state**.
- The other router will respond with a LSU (Link State Update) with the requested information.



When router Nancy requested information about 2.2.2.0 /24 it used a LSR. Router Donna will send the LSU with the information. Router Nancy will send an acknowledgment using a LSAck packet to finish. We are now in the **full state.** Both routers have a synchronized LSDB and we are ready to route!

Wonder what this looks like on a real router? Doing a **debug ip ospf adj** is faster than drawing all those pictures in Microsoft Visio:

```
Donna#debug ip ospf adj
OSPF adjacency events debugging is on
```

```
Donna#clear ip ospf process
Reset ALL OSPF processes? [no]: yes
```

Let me show you the debug:

```
Donna#
OSPF: Interface Loopback0 going Down
OSPF: 2.2.2.2 address 2.2.2.2 on Loopback0 is dead, state DOWN
OSPF: Interface FastEthernet0/0 going Down
OSPF: 2.2.2.2 address 192.168.12.2 on FastEthernet0/0 is dead, state DOWN
OSPF: Neighbor change Event on interface FastEthernet0/0
OSPF: DR/BDR election on FastEthernet0/0
OSPF: Elect BDR 0.0.0.0
OSPF: Elect DR 1.1.1.1
OSPF: Elect BDR 0.0.0.0
OSPF: Elect DR 1.1.1.1
       DR: 1.1.1.1 (Id)   BDR: none
OSPF: 1.1.1.1 address 192.168.12.1 on FastEthernet0/0 is dead, state DOWN
%OSPF-5-ADJCHG: Process 1, Nbr 1.1.1.1 on FastEthernet0/0 from FULL to
DOWN, Neighbor Down: Interface down or detached
OSPF: Neighbor change Event on interface FastEthernet0/0
OSPF: DR/BDR election on FastEthernet0/0
OSPF: Elect BDR 0.0.0.0
OSPF: Elect DR 0.0.0.0
       DR: none    BDR: none
OSPF: Remember old DR 1.1.1.1 (id)
OSPF: Interface Loopback0 going Up
OSPF: Interface FastEthernet0/0 going Up
OSPF: 2 Way Communication to 1.1.1.1 on FastEthernet0/0, state 2WAY
OSPF: Backup seen Event before WAIT timer on FastEthernet0/0
OSPF: DR/BDR election on FastEthernet0/0
OSPF: Elect BDR 2.2.2.2
OSPF: Elect DR 1.1.1.1
OSPF: Elect BDR 2.2.2.2
OSPF: Elect DR 1.1.1.1
       DR: 1.1.1.1 (Id)   BDR: 2.2.2.2 (Id)
OSPF: Send DBD to 1.1.1.1 on FastEthernet0/0 seq 0x1E09 opt 0x52 flag 0x7
len 32
OSPF: Rcv DBD from 1.1.1.1 on FastEthernet0/0 seq 0x886 opt 0x52 flag 0x7
len 32  mtu 1500 state EXSTART
OSPF: First DBD and we are not SLAVE
OSPF: Rcv DBD from 1.1.1.1 on FastEthernet0/0 seq 0x1E09 opt 0x52 flag 0x2
len 72  mtu 1500 state EXSTART
OSPF: NBR Negotiation Done. We are the MASTER
OSPF: Send DBD to 1.1.1.1 on FastEthernet0/0 seq 0x1E0A opt 0x52 flag 0x1
len 32
OSPF: Rcv DBD from 1.1.1.1 on FastEthernet0/0 seq 0x1E0A opt 0x52 flag 0x0
len 32  mtu 1500 state EXCHANGE
OSPF: Exchange Done with 1.1.1.1 on FastEthernet0/0
OSPF: Send LS REQ to 1.1.1.1 length 24 LSA count 2
OSPF: Rcv LS UPD from 1.1.1.1 on FastEthernet0/0 length 108 LSA count 2
OSPF: Synchronized with 1.1.1.1 on FastEthernet0/0, state FULL
```

```
%OSPF-5-ADJCHG: Process 1, Nbr 1.1.1.1 on FastEthernet0/0 from LOADING to
FULL, Loading Done
```

I highlighted some of the fields: you can see the 2 way communication, the struggle for power to determine who will be master or slave, exchange of the LSDB summary using another DBD packet and finally a LSQ and LSU.

And that's the end of this chapter. By now you have learned what OSPF is, how it forms neighbors adjacencies and the different packets it uses.

Do you enjoy reading this sample of How to Master CCNP ROUTE ?

Click on the link below to get the full version.

Get How to Master CCNP ROUTE Today