how to master CCNP TSHOOT

René Molenaar

Introduction

One of the things I do in life is work as a Cisco Certified System Instructor (CCSI) and after teaching 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 exam.

CCNP is one of the well-known certifications you can get in the world of IT. Cisco is the largest supplier of networking equipment but also famous for its CCNA, CCNP and CCIE certifications. Whether you are new to networking or already in the field for some time, getting a certification is the best way to prove your knowledge on paper! Having said that, I also love routing & switching because it's one of those fields in IT that doesn't change much…some of the protocols you are about to learn are 10 or 20 years old and still alive and kicking!

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 CCNP!

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://www.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.

Enjoy reading my book and good luck getting your CCNP 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. Network Maintenance and Troubleshooting methods

In this first chapter we will first look at some maintenance methods for networks. There are different models that will help you to maintain your networks and make your life easier. In the second part of this chapter we will look at some theoretical models that will help you with troubleshooting.

If you want to jump right into the technical action and start troubleshooting you might want to skip this chapter for now and get back to it later. However, on your CCNP TSHOOT exam you might encounter a couple of questions regarding network maintenance models and troubleshooting techniques so I do recommend you to read this chapter sometime.

Having said that, let's start talking about network maintenance! Network maintenance basically means you have to do what it takes in order to keep a network up and running and it includes a number of tasks:

- Troubleshooting network problems.
- Hardware and software installation/configuration.
- Monitoring and improving network performance.
- Planning for future network growth.
- Creating network documentation and keeping it up-to-date.
- Ensuring compliance with company policies.
- Ensuring compliance with legal regulations.
- Securing the network against all kind of threats.

Of course this list could be different for each network you work on and perhaps you are only responsible for a number of these tasks. All these tasks can be performed in the following way:

1. **Structured tasks.**
2. **Interrupt-driven tasks.**

Structured means you have a pre-defined plan for network maintenance that will make sure that problems are solved before they occur. As a network engineer this will also make your life a whole lot easier. **Interrupt-driven** means you just wait for trouble to occur and then fix it as fast as you can. Interrupt-driven is more like the “fireman” approach...you wait for trouble to happen and then you try to fix the problem as fast as you can. A structured approach where you have a network maintenance strategy and plan reduces downtime and it's more cost effective.

Of course you can never completely get rid of interrupt-driven tasks because sometimes things “just go wrong” but with a good plan we can reduce the number of interrupt-driven tasks for sure.

You don't have to think of a complete network maintenance model yourself; there are a number of well-known network maintenance models that we use. It's best to use one of the models that is best suited for your organization and adjustments if needed.

Here are some of the well-known network maintenance models:

- **FCAPS**:
  - Fault management.
  - Configuration management.
  - Accounting management.
  - Performance management.
  - Security management.

The FCAPS network maintenance model was created by the ISO (International Organization of Standardization).

- **ITIL**: IT Infrastructure Library is a set of practices for IT services management that focuses on aligning IT services with the needs of business.
- **TMN**: Telecomunications Management Network is another maintenance model that was created by the ITU-T (Telecommunications Standartization Sector) and is a variation of the FCAPS model. TMN targets management of telecommunications networks.
- **Cisco Lifecycle Services:** Of course Cisco has it's own network maintenance model which defines the different phases in the life of a Cisco network:
  - Prepare
  - Plan
  - Design
  - Implement
  - Operate
  - Optimize

If you decide to study CCDA (Cisco Certified Design Associate) you will learn a lot about the Cisco lifecycle which is also known as PPDIOO (Prepare, Plan, Design, Implement, Operate and Optimize).

Choosing which network maintenance model you will use depends on your network and the business. You can also use them as a template to create your own network maintenance model.

To give you an idea what a network maintenance model is about and what it looks like, here's an example for FCAPS:

- Fault management: we will configure our network devices (routers, switches, firewalls, servers, etc.) to capture logging messages and send them to an external server. Whenever an interface goes down or the CPU goes above 80% we want to receive an e-mail so we can see what is going on.
- Configuration management: Any changes made to the network have to be logged. We will use a change management so relevant personnel will be notified of planned network changes. Changes to network devices have to be reported and acknowledged before they are implemented.
- Accounting management: We will charge (guest) users for usage of the wireless network so they'll pay for each 100MB of data or something. It's also commonly used to charge people for long distance VoIP calls.
- Performance management: Network performance will be monitored on all LAN and WAN links so we know when things go wrong. QoS (Quality of Service) will be configured on the appropiate interfaces.

- Security management: We will create a security policy and implement it by using firewalls, VPNs, intrusion prevention systems and use AAA (Authorization, Authentication and Accounting) servers to validate user credentials. Network breaches have to be logged and a appropiate response has to be made.

You can see FCAPS is not just a "theoretical" method but it truly describes "what", "how" and "when" we will do things.

Whatever network maintenance model you decide to use, there are always a number of routine maintenance tasks that should have listed procedures, here are a couple of examples:

- **Configuration changes**: Business are never static but they change all the time. Sometimes you need to make changes to the network to allow access for guest users, normal users might move from one office to another so you'll have to make changes to the network to facilitate this.
- **Replacement of hardware**: Older hardware has to be replaced with more modern equipment and it's also possible that production hardware fails so we'll have to replace it immediately.
- **Backups**: If we want to recover from network problems such as failing switches or routers then we need to make sure we have recent backups of configurations. Normally you will use scheduled backups so you will save the running-configuration each day, week, month or whatever you like.
- **Software updates**: We need to keep our network devices and operating systems up-to-date. Bugs are fixed but also to make sure we don't have devices that are running older software that has security vulnerabilities.
- **Monitoring**: We need to collect and understand traffic statistics and bandwidth utilization so we can spot (future) network problems but also so we can plan for future network growth.

Normally you will create a list with the tasks that have to be done for your network. These tasks can be assigned a certain priority. If a certain access layer switch fails then you will likely want to replace it as fast as you can but a failed distribution or core layer device will have a much higher priority since it impacts more users of the network. Other tasks like backups and software updates can be scheduled. You will probably want to install software updates outside of business operating hours and backups can be scheduled to perform each day after midnight or something. The advantage of scheduling certain tasks is that network engineers will less likely forget to do them.

Making changes to your network will sometimes impact productivity of users who rely on the network availability. Some changes will have a huge impact, changes to firewalls or access-list rules might impact more users then you'd wish for. For example you might want to install a new firewall and planned for a certain result. Accidentally you forgot about a certain application that uses random port numbers and you end up troubleshooting this issue. Meanwhile some users are not able to use this application (and shouting at you while you try to fix it...;).

Larger companies might have more than 1 IT department and each department is responsible for different network services. If you plan to replace a certain router tommorow at 2AM then you might want to warn the "Microsoft Windows" guys department because their servers will be unreachable. You can use change management for this. When you plan to make a certain change to the network then other departments will be informed and they can object if there is a conflict with their planning.

When you want to implement change management you might want to think about the following:

- Who will be responsible for authorizing changes to the network?
- Which tasks will be performed during scheduled maintenance windows?
- What procedures have to be followed before making a change? (for example: doing a "copy run start" before making changes to a switch).
- How will you measure the success or failure of network changes? (for example: if you plan to change a number of IP addresses you will plan the time required to make this change. If it takes 5 minutes to reconfigure the IP addresses and you end up troubleshooting 2 hours because something else is not working you might want to "rollback" to the previous configuration. How much time do you allow for troubleshooting? 5 minutes? 10 minutes? 1 hour?
- How, when and who will add the network change to the network documentation?
- How will you create a rollback plan so you can restore a configuration to the previous configuration in case of unexpected problems?
- What circumstances will allow change management policies to be overruled?

Another task we have to do is to create and update our network documentation. Whenever a new network is designed and created it should be documented.
The more challenging part is to keep it up-to-date in the future. There are a number of items that you should find in any network documentation:

- Physical topology diagram: This should show all the network devices and how they are physically connected to each other.
- Logical topology diagram: This should show how everything is connected to each other. Protocols that are used, VLAN information etc.
- Interconnections: It's useful to have a diagram that shows which interfaces of one network device are connected to the interface of another network device.
- Inventory: You should have an inventory of all network equipment, vendor lists, product numbers, software versions, software license information and each network device should have an organization tag assess number.
- IP Addresses: You should have a diagram that covers all the IP addresses in use on the network and on which interfaces they are configured.
- Configuration management: Before changing a configuration we should save the current running-configuration so it's easy to restore to a previous (working) version. It's even better to keep an archive of older configurations for future use.
- Design documents: Documents that were created during the original design of the network should be kept so you can always check why certain design decisions were made.

It's also a good idea to work with step-by-step guidelines for troubleshooting or using templates for certain configurations that all network engineers agree on to use, here are some examples to give you an idea:

```
interface FastEthernet0/1
 description AccessPoint
 switchport access vlan 2
 switchport mode access
 spanning-tree portfast
```

Here's an example for access interfaces connected to wireless access points. Portfast has to be enabled for spanning-tree, the access points have to be in VLAN 2 and the switchport

has to be changed to "access" manually.

```
interface FastEthernet0/2
 description VOIP
 interface FastEthernet0/2
 description ClientComputer
 switchport access vlan 6
 switchport mode access
 switchport port-security
 switchport port-security violation shutdown
 switchport port-security maximum 1
 spanning-tree portfast
 spanning-tree bpduguard enable
```

Here's a template for interfaces that connect to client computers. The interface has to be configured for "access" mode manually. Port security has to be enabled so only 1 MAC address is allowed (the computer). The interface has to go into forwarding mode immediately so we configure spanning-tree portfast and if we receive a BPDU the interface should go into err-disabled. Working with pre-defined templates like there will reduce the number of errors because everyone agrees on the same configuration. If you give each network engineer instructions to "protect the interface" you'll probably end up with 10 different configurations…

```
interface GigabitEthernet0/1
 description TRUNK
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport trunk nonegotiate
```

Here's one more example for trunk links. If you tell 2 network engineers to "configure a trunk" you might end up with one interface configured for 802.1Q encapsulation and the other one for ISL encapsulation. If one network engineer disabled DTP and the other one configure the interface as "dynamic desirable" then it will also fail. If you instruct them to configure a trunk according to a template then we'll have the same configuration on both sides.

Enough about network maintenance, in the second part of this chapter we'll take a look at the theory of troubleshooting.

There are different reasons why things go wrong on our networks, humans make errors in their configurations, hardware can fail, software updates may include bugs and changing traffic patterns might cause congestion on our networks. To troubleshoot these errors there are different approaches and some are more effective than others.

Troubleshooting consists of 3 steps:

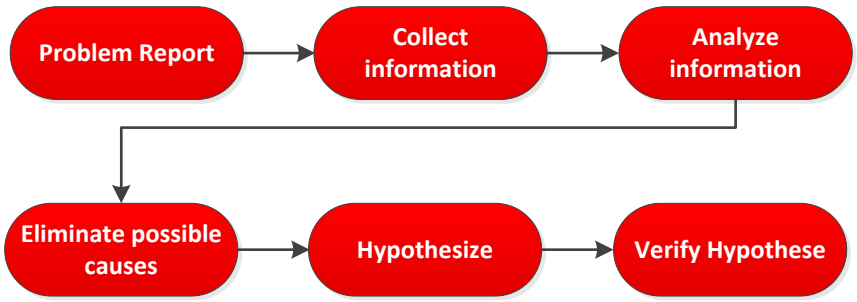
It all starts when someone or something **reports a problem**. Often this will be a user that calls the helpdesk because something is not working as expected but it's also possible that you find issues because of network monitoring (you do monitor your network right? ☺). The

next step is to **diagnose** the problem and it's important to find the root of the problem. Once you have found out the problem you will implement a (temporary) solution.

Diagnosing the problem is one of the most important steps to do because we need to find the root cause of the problem, here's what we do to diagnose the problem:

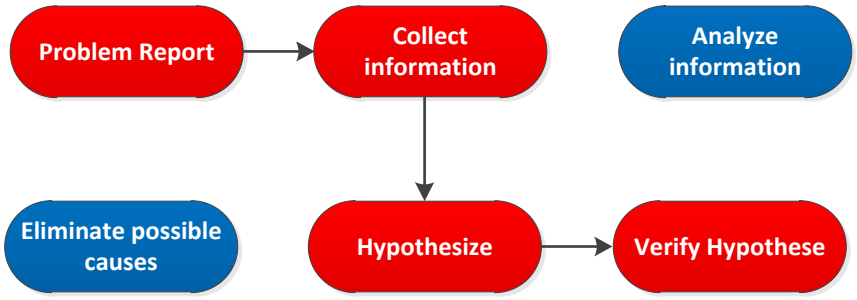
- **Collect information**: Most of the time a problem report doesn't give us enough information. Users are very good at reporting "network is down" or "my computer doesn't work" but this doesn't tell us anything. We need to collect information by asking our users detailed questions or we use network tools to gather information.
- **Analyze information**: Once we have gathered all information we will analyze it so see what is wrong. We can compare our information to previously collected information or other devices with similar configurations.
- **Eliminate possible causes**: We need to think about the possible causes and eliminate the potential causes for the problem. This requires thorough knowledge of the network and all the protocols that are involved.
- **Hypothesize**: After eliminating possible causes you will end up with a couple of possible causes that could be the problem. We will select the most likely cause for the problem.
- **Verify hypothesis**: We will test our hypothesis to see if we are right or wrong. If we are right we have a victory…if we are wrong we test our other possible causes.

If you don't use a structured approach for troubleshooting you might just "follow your gut feeling" and get confused because you forget what you already tried or not. It's also easier if you work together with other network engineers because you can share the steps you already went through.



Here are the steps in a nice flowchart; we call this the **structured troubleshooting approach**. However if you have a lot of experience with the network you are working on and as you become better at troubleshooting this approach might be too time-consuming.

Instead of walking through all the different steps in the structured troubleshooting approach we can also jump from the "collect information" step directly to the "hypothesize" step and skip the "analyze information" and "eliminate possible causes" steps If you are inexperienced with troubleshooting it's best to use the structured troubleshooting approach. As you become better at troubleshooting you might want to skip some of the steps…we call this the **shoot from the hip** approach.



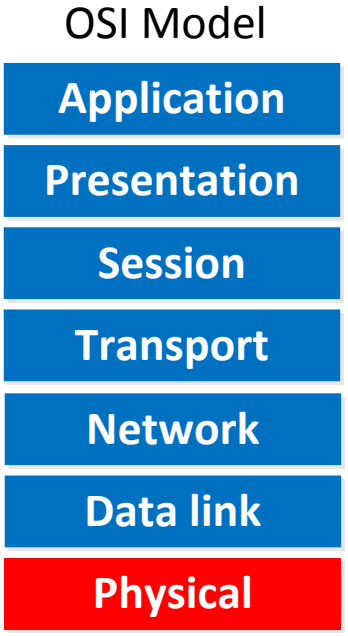
Here's the shoot from the hip model. The steps that we skip are in blue. If your instincts are wrong you won't lose your life but you will lose valuable time.
If you are right however you'll save a lot of time (or become the new sheriff in town).

Eliminating possible causes is an important step in the troubleshooting process and there are a couple of approaches how you can do this, here they are:

- **Top-down.**
- **Bottom-up.**
- **Divide and conquer.**
- **Follow the traffic path.**
- **Spot the difference.**
- **Replace components.**

Let's walk through the different approaches one-by-one!

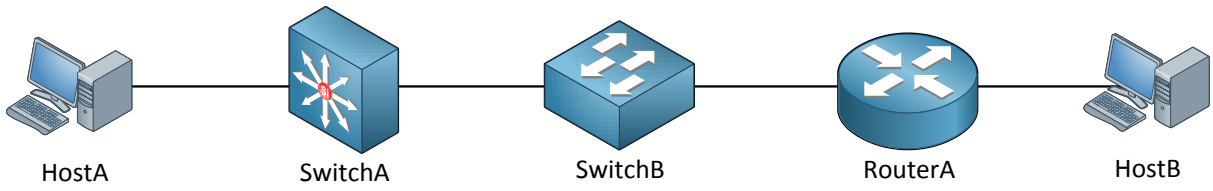
Top-down means we start at the top of the OSI model (application layer) and work our way further down to the bottom. The idea is that we will check the application to see if it's working and assume that if a certain layer is working that all the layers below are also working. If you send a ping from one computer to another (ICMP) you can assume that layer 1,2 and 3 are operational. The downside of this approach is that you need access to the application that you are troubleshooting.



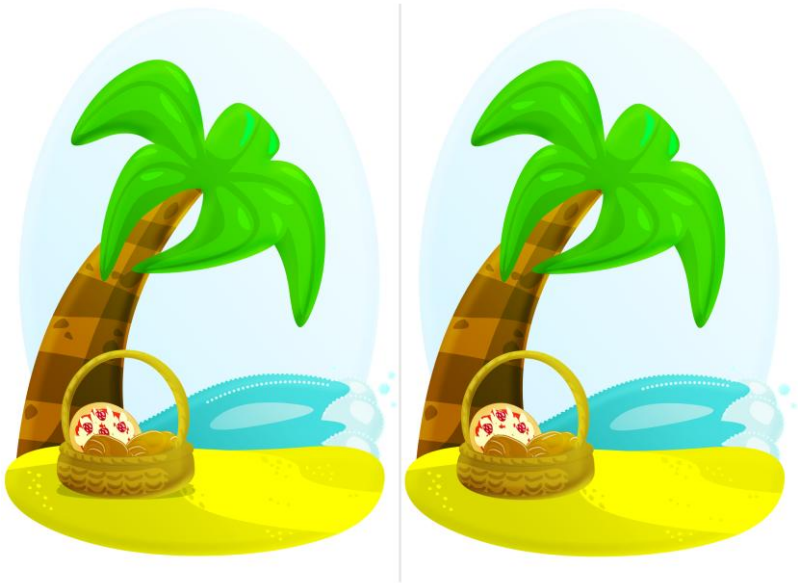
Bottom-up means we start at the bottom of the OSI model and we'll work our way up. We will start with the physical layer which means we check our cables and connectors, move up to the data link layer to see if Ethernet is working, Spanning-tree is working ok, port security is not causing issue, VLANs are configured properly and then move onto the network layer. Here we will check our IP addresses, access-lists, routing protocols and so on. This approach is very thorough but also time-consuming. If you are new to troubleshooting I would recommend to use this method because you will eliminate all the possible causes for problems.

Divide and conquer means we start in the middle of the OSI-model. You can use this model if you are not sure if top-down or bottom-up are more effective. The idea is that you'll try to send a ping from one device to another. If the ping works you know that layer 1-3 are operational and you can work your way up in the OSI model.

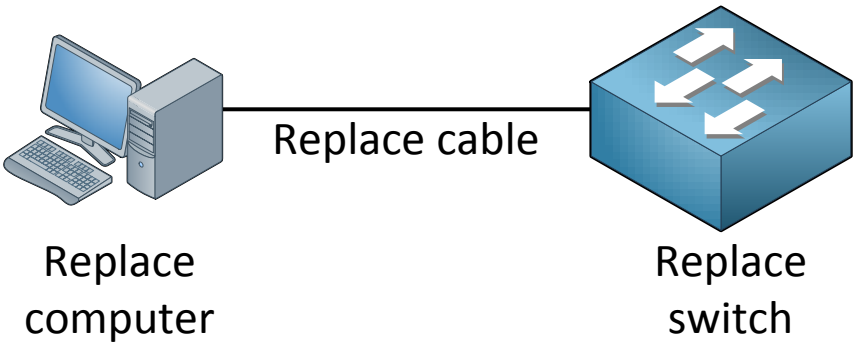
If the ping fails you know something is not right and you'll work your way to the bottom of the OSI model.



The **follow the traffic path** is very useful. First we'll try to send a ping from HostA to HostB. If it fails we'll check all the devices in its path. First we'll verify if SwitchA is configured correctly, it it's looking good we'll move onto SwitchB, verify it and then move onto RouterA.



You've probably done one of these before. **Spotting the difference** in configurations or the output of show commands can be useful but it's very easy to miss something. If you have a number of branch routers with a similar configuration and only one is not working you can see if there's a difference in the configuration. Network engineers that don't have a lot of experience usually use this approach. You might be able to solve the problem but there's a risk that you don't really know what you are doing.



The last approach to solve our problem is to **replace components**. Let's say we have a scenario where a computer is unable to access the network. In the example above I could replace the computer to eliminate any chance of the computer being the problem. We could replace the cable and if we suspect the switch we can replace it with a new one and copy the old configuration to see if there are any hardware problems.

This is all the theory I wanted to share with you about network maintenance and troubleshooting.
We can talk all day long about different methods and such but the key to becoming an expert with troubleshooting consists of two things:

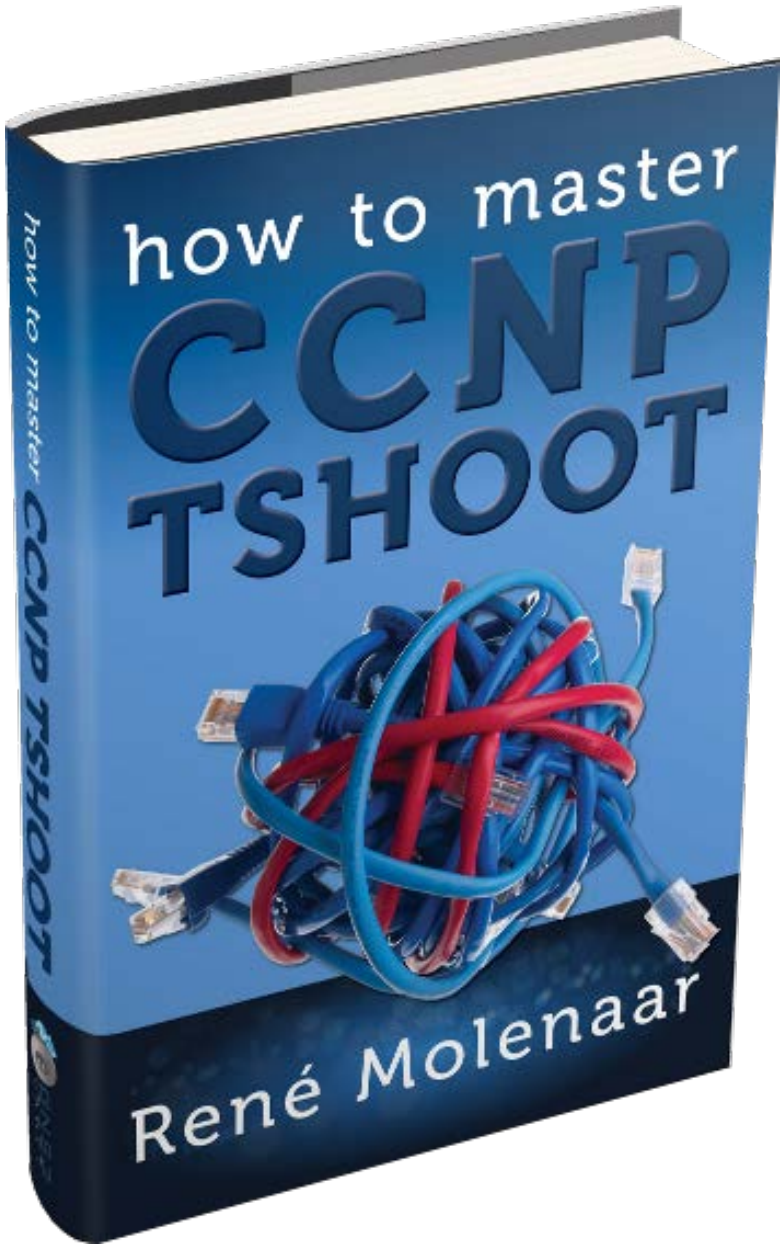
- Truly **understand** all the different networking protocols like OSPF, EIGRP, BGP, spanning-tree and everything else you learned in CCNA and CCNP ROUTE/SWITCH. You can't fix something if you have no idea how it works.
- **Gain experience by doing labs** and **troubleshoot** broken networks!

You can read all the books about driving a car, how an engine works, what a clutch is or how to use your mirrors but at the end of the day…you have to **sit in a car** and **start driving** to learn how to drive a car. In the upcoming chapters of this book you will learn about all the different things that could go wrong with all the protocols and I will teach you how to troubleshoot them. I will also refer to some of the troubleshooting labs on GNS3Vault that I created for you to practice!

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

Click on the link below to get the full version.

Get How to Master CCNP TSHOOT Today



2. Tools for Troubleshooting

Before we start looking at the troubleshooting of the different protocols I want to give you an overview of all the different tools you can use for troubleshooting. Some of these might be familiar to you already but some are probably new to you. In this chapter we'll walk through all the different tools you can use and I'll give you some examples how to use them.

"If I had eight hours to chop down a tree, I'd spend six hours sharpening my ax"
~Abraham Lincoln

The first item we'll look at is NTP. Once you start working with logging information and debug output you need to make sure they show the correct date and time or your information will be useless. Cisco devices can be configured to use an external NTP server to synchronize their clocks. You can also configure most Cisco devices to become a NTP server.

```
Router(config)#ntp server pool.ntp.org
```

First I configure an external NTP server with the **ntp server** command. I've chosen to use one of the NTP servers from ntp.org.

```
Router#show ntp status
Clock is synchronized, stratum 3, reference is 217.121.137.227
nominal freq is 250.0000 Hz, actual freq is 250.0000 Hz, precision is 2**18
reference time is D35E243C.A5B38A7C (13:03:56.647 UTC Wed May 16 2012)
clock offset is -10.1544 msec, root delay is 29.10 msec
root dispersion is 52.75 msec, peer dispersion is 2.40 msec
```

If everything is OK it should show you that the clock is now synchronized. We still have to configure the correct time zone.

```
Router(config)#clock timezone CET +1
```

I configured my time zone to be CET (European Time Zone). You can type in whatever name you want for the time zone, it doesn't matter. It's important that you configure the correct offset however.

In Europe we have a summer time which means we turn the clock one hour forward on the last Sunday in March from 02:00 to 03:00. On the last Sunday of October we turn the clock backwards from 03:00 to 02:00. This is something we'll have to configure on our Cisco device.

```
Router(config)#clock summer-time CETrecurring las sun mar 02:00 last sun
oct 03:00
```

This takes care of the clock configuration. It's important that we enable the date,time and timezone in our debug and logging information. When you store logging or debug

information it's important that they have the correct date and time on them.

```
Router(config)#service timestamps debug datetime msec localtime show-
timezone year
Router(config)#service timestamps log datetime msec localtime show-timezone
year
```

Use the **service timestamps** command to add the correct date and time to debug and logging information.

```
May 16 2012 15:16:23.214 CET: RIP: sending v2 flash update to 224.0.0.9 via
FastEthernet0/0 (192.168.81.154)
May 16 2012 15:16:23.214 CET: RIP: build flash update entries
May 16 2012 15:16:23.214 CET:   1.1.1.0/24 via 0.0.0.0, metric 1, tag 0
May 16 2012 15:16:23.218 CET: RIP: sending v2 flash update to 224.0.0.9 via
Loopback0 (1.1.1.1)
```

This is an example of a debug. You can see it adds the correct date, time and time zone. It's a good idea to configure one or two devices in your network to synchronize with an external NTP server. You can configure a Cisco device to become a NTP server for the rest of the devices in your network. If you want an example how to do this I can recommend you to try this lab:

http://gns3vault.com/Network-Management/ntp-network-time-protocol.html

Now our clocks are working correctly we can take a look at logging. Everything that happens on your router or switch can be logged. We have different levels of importance for logging information. By default you'll see the logging information on your console, like this one:

```
May 16 2012 15:24:53.080 CET: %LINK-5-CHANGED: Interface FastEthernet0/0,
changed state to administratively down
May 16 2012 15:24:54.080 CET: %LINEPROTO-5-UPDOWN: Line protocol on
Interface FastEthernet0/0, changed state to down
```

This is an example of an interface that's going down.

```
May 16 2012 15:25:19.893 CET: %SYS-5-CONFIG_I: Configured from console by
console
```

You have probably seen this one before; you'll see it when you exit the global configuration mode. There are different severity levels for logging information. An interface that goes down is probably more important to know than a message that tells us we exited the global configuration.

Here are the severity levels:

0. Emergencies
1. Alerts
2. Critical
3. Errors
4. Warnings
5. Notifications
6. Informational
7. Debugging

By default you'll see all of these messages on the console. If you don't want to see everything you can change this behavior:

```
Router(config)#logging console errors
```

For example you can use configure the **logging console** command so it only shows you severity level 3 (errors) and lower. The message about the interface that was going down is a **notification** in case you were wondering.

```
Router#show logging history
Syslog History Table:1 maximum table entries,
saving level warnings or higher
 29 messages ignored, 11 dropped, 0 recursion drops
 2 table entries flushed
 SNMP notifications not enabled
   entry number 3 : LINK-3-UPDOWN
    Interface FastEthernet0/0, changed state to up
    timestamp: 1688
```

We can use the **show logging history** command to see the logging history of this Cisco router. It doesn't store everything. You can see it says "saving level warnings or higher". This logging information is saved in the RAM of your device. Once you reboot it you will lose this logging history.

```
Router(config)#logging buffered 4096
```

You can change the size of the buffer if needed. As soon as the buffer is full old logging information will be discarded. In the example above we have 4096 bytes we can use for logging information.

It's not a good idea to store logging information locally on your device. One reboot and you'll lose valuable information. It's best to use an external server for this.

```
Router(config)#logging 192.168.1.100
```

Use the **logging** command to set the IP address for your logging server. All logging information will be sent towards this server with the exception of debugging (level 7) messages by default.

Normally you probably only want to see debug information on your console or telnet/SSH session but you can store it in on your logging server too if you want:

```
Router(config)#logging trap 7
```

You need to use the **logging trap** command to 7 so it will also store debug information on your logging server.

Besides syslog there is another method to store logging information to an external server. **SNMP (Simple Network Management Protocol)** can be used to collect statistics from network devices including Cisco routers and switches.

SNMP consists of 2 items:

- **NMS (Network Management System)**
- **SNMP Agents**

The NMS is the external server where you want to store logging information. The SNMP agents run on the network devices that we want to monitor. The NMS can query a SNMP agent to collect information from the network device. SNMP has multiple versions, the most popular ones being:

- SNMP version 2c
- SNMP version 3

SNMP version 3 offers security through authentication and encryption which SNMP version 2c does not. SNMP version 2c however is still more common. Let me show you a simple example for SNMP version 2c:

```
Router(config)#snmp-server community TSHOOT ro
```

First we'll have to configure a **community string**. Think of this as a password that the SNMP agent and NMS have to agree upon. I called mine "TSHOOT". The **ro** stands for **read-only**. SNMP isn't just for retrieving information; we can also use it to instruct our routers and switches to perform an action.

```
Router(config)#snmp-server location Amsterdam GNS3Vault Lab
Router(config)#snmp-server contact info@gns3vault.com
```

These two steps are not required but it's useful to specify a **location** and **contact.** This way you'll at least know where the device is located whenever you receive information through SNMP.

```
Router(config)#snmp-server host 192.168.12.2 version 2c TSHOOT
```

The messages that the SNMP agent sends to the NMS are called **SNMP traps**. Of course we want to send these to an external server so I'll configure the IP address of the SNMP server. I also have to specify the version and the community string.

```
Router(config)#snmp-server enable traps
```

Last but not least I'll have to enable the SNMP traps. If I use the **snmp-server enable traps** command it will enable **all SNMP traps**.

```
Router#show run | include traps
snmp-server enable traps snmp authentication linkdown linkup coldstart
warmstart
snmp-server enable traps vrrp
snmp-server enable traps ds1
snmp-server enable traps tty
snmp-server enable traps eigrp
snmp-server enable traps casa
snmp-server enable traps xgcp
snmp-server enable traps bulkstat collection transfer
snmp-server enable traps isdn call-information
snmp-server enable traps isdn layer2
```

This is only a portion of everything that you'll see in the running-configuration. This is a great way to test SNMP but on a production network it's better to take a look at the different traps and only enable the ones you feel are necessary. One of the SNMP traps in the example above is related to EIGRP. If anything happens with the EIGRP routing protocol a SNMP trap will be send towards the SNMP server.

If you never tried saving syslog information to an external server or played with SNMP I recommend trying "Solarwinds Kiwi Syslog server". You can download a 30 day trial and practice the commands above yourself:

http://www.kiwisyslog.com

Let's continue by looking at some methods how we can save our configs.

```
Router#copy running-config startup-config
Destination filename [startup-config]?
```

You are probably familiar with this command…if not, shame on you! ☺ This is the simplest method of copying your running configuration to the startup-config file which is stored in your nvram. It's a good method to store your configuration but you'll lose your configuration if your device has a hardware failure.

It's a good idea to have a backup server somewhere in your network; this could be something as simple as running a FTP or TFTP server on your own computer. Tftpd32 is a nice free windows utility for running a TFTP server. You don't even have to install it, it runs as an executable:

http://tftpd32.jounin.net/

We can use the copy command to copy our configuration and/or files on the flash to a wide range of devices, for example:

```
Router#copy running-config ftp://username:password@192.168.1.1/myconfig.cfg
Address or name of remote host [192.168.1.1]?
Destination filename [myconfig.cfg]?
```

If you studied CCNA you are probably familiar with using the copy command like this. The command above will help you to copy your running-config to a FTP server. The downside of this method is that you have to type in the FTP username and password yourself.

```
Router(config)#ip ftp username myusername
Router(config)#ip ftp password mypassword
```

If you want you can save the FTP username and password in your configuration. By default it will be saved in cleartext. You can use the "service password-encryption" command to encrypt it using the Cisco type 7 proprietary encryption but don't forget that this can be easily decrypted!

```
Router#copy http://www.gns3vault.com/config.txt flash:/ configs/config.txt
```

You can also use HTTP to copy files from a webserver to your device.

FTP and HTTP are easy methods to copy configuration files and/or IOS images from and to your device but everything will be sent in **clear-text**. You can also use HTTPS or SCP to copy so traffic will be encrypted!

You've probably played with the copy command before but do you also know there's a method to archive configurations on your Cisco device? Let me show you how it works!

```
Router(config)#archive
Router(config-archive)#path ftp:/archive/$h-config
Router(config-archive)#write-memory
Router(config-archive)#time-period 60
```

First you need to use the **archive** command to get into the archive configuration. I configured my router to create an archive on the FTP server and the configuration files should be saved in the "archive" folder. The filenames will be appended with the hostname of the device because I used the **$h** parameter in front of the filename.

Every time someone copies the running configuration to the NVRAM the device will store a copy on the FTP server because of the **write-memory** command. The **time-period** command can be used to configure a schedule. Every 60 minutes the running-configuration will be "archived" to my FTP server. 60 minutes is very short and on a production network you probably want to configure this to 1440 minutes (24 hours) or maybe 10080 minutes (1 week).

```
Router#show archive
The next archive file will be named ftp:/archive/Router-config-5
 Archive #  Name
   0      ftp:/archive/Router-config-1
   1      ftp:/archive/Router-config-2
   2      ftp:/archive/Router-config-3
   3      ftp:/archive/Router-config-4
```

You can use the **show archive** command to check how many backup configurations you have stored. In my example there are 4 backup configurations and the next one will be saved as "Router-config-5". It's important to have multiple backups of configuration files instead of just saving the "last" one. In case of trouble or misconfiguration you can always rollback to one of the earlier configurations that you archived.

Let's say someone changed the running-configuration on my device, made some errors and I want to roll back to a previous version. What would be the best way to do this? Of course I could do a reload and hope that they didn't overwrite the startup-config.
The downside is that a reload could take a couple of minutes and it will disrupt network services. Anything else we can do?

```
Router#copy startup-config running-config
```

If the startup-config is still good we could use the copy command. If you studied CCNA you might recall that copying something to the running-config will **not replace** it but the two configurations will **merge**.

There is a command however that lets you replace the running-configuration **without rebooting** your device.

```
Router#configure replace ftp:/archive/Router-config-1 list
This will apply all necessary additions and deletions
to replace the current running configuration with the
contents of the specified configuration file, which is
assumed to be a complete configuration, not a partial
configuration. Enter Y if you are sure you want to proceed. ? [no]: yes

!List of Commands:
no hostname Router
hostname MyRouter
end

Total number of passes: 1
Rollback Done

MyRouter#
```

We can use the **configure replace** command to do this. I will use one of my archived configurations on the FTP server to replace my running-configuration. The **list** parameter will show me all the commands that are different between the archived configuration and the running-configuration. In my example I'm using the first archived configuration file to copy it into the running-configuration. The only difference is the hostname of my router which has changed from "Router" to "MyRouter". This is a clean method of restoring the running-configuration without reloading your device.

Enough about configurations, there are more tricks that you should be familiar with in order to become a good troubleshooter. I want to show you some techniques for "show" commands that will be helpful.

```
Router#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, 00:00:16, FastEthernet0/0
```

Let's start with **show ip route**. Let's imagine this router receives an IP packet destined for IP address 2.2.2.2 and I want to see if it knows where to send it. I can type "show ip route" and browse through the routing table myself.

This will work but I'll have to scan the table myself.

```
Router#show ip route 2.2.2.2
Routing entry for 2.2.2.0/24
  Known via "eigrp 1", distance 90, metric 156160, type internal
  Redistributing via eigrp 1
  Last update from 192.168.12.2 on FastEthernet0/0, 00:02:12 ago
  Routing Descriptor Blocks:
  * 192.168.12.2, from 192.168.12.2, 00:02:12 ago, via FastEthernet0/0
      Route metric is 156160, traffic share count is 1
      Total delay is 5100 microseconds, minimum bandwidth is 100000 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 1
```

I can also type in **show ip route 2.2.2.2.** This will scan the routing table and it will show me the network that matches the IP address that I'm looking for. In the example above you can see that 2.2.2.0/24 matches the IP address that I was looking for. It also gives me more detailed information.

```
Router#show ip route 3.3.3.3
% Network not in table
```

If you are looking for an IP address that does not match any networks you'll get a nice message that the network is not in your routing table. However don't be too fast with drawing conclusions. If you have a default route in your routing table it won't show up as the network for this IP address, let me show you what I mean:

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

Let's create a default route; I don't care what it is I just want something in my routing table.

```
Router#show ip route

Gateway of last resort is 192.168.12.2 to network 0.0.0.0

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, 00:16:44, FastEthernet0/0
S*   0.0.0.0/0 [1/0] via 192.168.12.2
```

You can see I now have a default route in my routing table.

```
Router#show ip route 3.3.3.3
% Network not in table
```

If I do a lookup for IP address 3.3.3.3 it will tell me there is no network that matches this IP address. It matches the default route however, keep in mind to check the routing table yourself to see if there is a default route or not.

Our "show ip route" command offers us a nice way to check a certain entry in the routing

table but not all show commands have similar options. There are some generic options that we can use for **all show** commands.

Show commands can be filtered by using the | character (it's called a pipe) and combining it with the **begin, section, include or exclude** parameters. Let's take a look at some examples!

```
Router#show run | begin router eigrp

router eigrp 1
 network 0.0.0.0
 no auto-summary
```

If I want to check my EIGRP configuration in the running-configuration I can type in "show run" and hammer the space or enter button to get to the EIGRP section of the configuration. *To make things easier* - I can also use the **begin** parameter. By using the "**show run | begin router eigrp**" command it will skip everything in the running configuration until the first line that matches "router eigrp".

```
 Router#show run | section vty
 line vty 0 4
 password vault
 login
```

We can also use the **section** keyword. In the example above I'm jumping straight to the "vty" section of the running configuration.

```
Router#show ip interface brief
Interface                  IP-Address      OK? Method Status
Protocol
FastEthernet0/0            192.168.12.1    YES manual up  up
Loopback0                  1.1.1.1         YES manual up  up
Loopback1                  unassigned      YES unset  administratively down
down
```

This is an example of the "show ip interface brief" command without any filters.

```
Router#show ip int brief | include administratively
Loopback1              unassigned      YES unset  administratively down
down
```

If I want a quick overview of all the interfaces with a "shutdown" command on them I can use the **"show ip int brief | include administratively"** command. This will **only** show the lines that have the word "administratively" in them.

```
Router#show ip int brief | include admini
Loopback1              unassigned      YES unset  administratively down
down
```

Of course you don't have to fully type what you are looking for; just typing "admini" will match "administratively" as well. This gives me a nice clean overview of what I'm looking for.

```
Router#show ip interface brief | exclude up
Interface                  IP-Address      OK? Method Status
Protocol
Loopback1                  unassigned      YES unset administratively down
down
```

I can also use the **exclude** keyword to filter all the lines that have "up" in them.

In the examples above I used the begin, section, include or exclude parameters and combined them with a word or sentence like "vty", "router eigrp" or "administratively". We can also use regular expressions. If you studied CCNP ROUTE you have seen regular expressions in combination with BGP.

```
Router#show ip int brief | include ^Loo
Loopback0                  1.1.1.1     YES manual up                    up
Loopback1                  unassigned  YES unset  administratively down
down
```

Here's an example of a regular expression. The ^ symbol means that the line has to start with whatever I type behind the ^ symbol. ^Loo means I want to see all lines that start with "Loo". In this case it will show me all the loopback interfaces.

```
Router#show processes | include ^CPU|ARP Input
CPU utilization for five seconds: 0%/0%; one minute: 0%; five minutes: 0%
  16 Mwe 605F6A3C           12          78     153 4732/6000    0 ARP Input
  91 Mwe 6068862C            0           1       0 5616/6000    0 RARP Input
```

A good example of the power of regular expressions is by demonstrating it with the "show processes" command. The ^CPU means I want to see all lines that start with "CPU". The "ARP Input" will show me all lines that contain "ARP Input". As a result I have a nice overview with just the ARP information.

```
Router#show processes | include ARP Input
  16 Mwe 605F6A3C           12          80     150 4732/6000    0 ARP Input
  91 Mwe 6068862C            0           1       0 5616/6000    0 RARP Input
```

If I don't add the ^CPU part then it will only show me the two lines containing "ARP Input" which isn't very useful…

Using the begin, include, exclude and section parameters makes my show commands far more useful. Right now I only see the output of the show command on my screen but it's also possible to redirect the output to a file.

```
Router#show tech-support | redirect tftp://192.168.12.100/techsupport.txt
```

If you ever required support from Cisco they might have asked for the "show tech-support" command. This show command produces a lot of output and instead of copy/pasting whatever you see on your screen it might be easier to save the output in a file. Using the **redirect** parameter we can copy the output to a (TFTP) server and save it as a file called "techsupport.txt".

The redirect parameter saves the output in a file and does **not show you the output on your screen**.

```
Router#show run | tee flash:myconfig.txt
Building configuration...

Current configuration : 714 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Router
```

The **tee** parameter does the same thing as the redirect parameter but it **does show** the output of the show command on my screen. In the example above I'm also saving the output on the flash in a file called "myconfig.txt".

```
Router#dir flash:
Directory of flash:/

    1  -rw-         775                    <no date>  myconfig.txt
```

Use the **dir flash:** command to see the contents of the flash.

```
Router#more flash:myconfig.txt
Building configuration...

Current configuration : 714 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Router
```

You can look at the contents of a file on your device with the **more** command. In the example above I'm looking at the contents of the myconfig.txt file.

```
Router#show tech-support | append flash:techsupport.txt
```

Redirect and tee will overwrite the old file if you use the same filename. We can also use **append** which is the same as redirect but it will add the output to the file instead of overwriting it. Not all file systems support append so it's possible that you get an error if you try this.

The last show commands I want to "show" you are the ones we use when we troubleshoot hardware-related issues.

```
route-views.optus.net.au>show processes cpu sorted 5min
CPU utilization for five seconds: 0%/0%; one minute: 2%; five minutes: 3%
 PID Runtime(ms)     Invoked      uSecs   5Sec   1Min   5Min TTY Process
 172   424120884     2295242     184788  0.00%  1.35%  1.58%   0 BGP
Scanner
   4   382944880    20406399      18766  0.00%  0.89%  1.24%   0 Check
heaps
 137       21640  2030982109          0  0.15%  0.12%  0.12%   0 HQF Shaper
Backg
 138       18472  2030982311          0  0.15%  0.10%  0.08%   0 HQF Input
Shaper
  88    17871980   267174489         66  0.00%  0.07%  0.07%   0 IP Input
 235    22159976   138390034        160  0.00%  0.06%  0.07%   0 BGP Router
  50    10475216     3398459       3082  0.00%  0.06%  0.05%   0 Net
Background
```

I used one of the looking glass servers from http://www.bgp4.as/looking-glasses to telnet to and check the CPU load with the **show processes** command. Looking glass servers are BGP routers with a partial or full internet routing table. You don't have enable access but you can use some of the show commands to look at. In my example I used the **cpu sorted 5min** parameter to sort the processes based on a 5 minute interval. The load of the CPU is only 3% on a 5 minute interval. BGP scanner is the process that uses 1.58% of the CPU on a 5 minute interval.

Normally you shouldn't see a very high CPU load on a Cisco router or switch. Switches use a ASIC (Application Specific Integrated Circuit) which means that the forwarding of frames is done using hardware tables, not in software (CPU). Routers nowadays use CEF (Cisco Express Forwarding) which uses specialized tables so we don't have to use the CPU for each IP packet entering the router.

```
route-views.optus.net.au>show memory
               Head    Total(b)     Used(b)     Free(b)   Lowest(b)  Largest(b)
Processor   641DBCA0   417479404   365773404    51706000     180220    16777164
      I/O    E000000    33554432     3920924    29633508   26613200    29273820
Transient   7D000000    16777216       10208    16767008   16110700    16766732
```

We can also verify the memory. Especially if you run BGP and have the full Internet routing table you'll need quite some memory. You can use the **show memory** command to verify this. If your router runs out of memory you will see a %SYS-2-MALLOCFAIL message on your console indicating that a process was unable to get enough memory.

Another thing you might want to check if you think your router or switch has hardware errors are your interfaces.

```
Router#show interface gi0/1
GigabitEthernet0/1 is up, line protocol is up
  Last clearing of "show interface" counters never
  Input queue: 0/1000/0/114 (size/max/drops/flushes); Total output drops: 4
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 12000 bits/sec, 15 packets/sec
  5 minute output rate 15000 bits/sec, 17 packets/sec
     299904796 packets input, 1017879192 bytes, 0 no buffer
     Received 1433615 broadcasts (0 IP multicasts)
     0 runts, 0 giants, 0 throttles
     10 input errors, 6 CRC, 2 frame, 0 overrun, 12 ignored
     0 watchdog, 35502363 multicast, 0 pause input
     366840355 packets output, 3958512366 bytes, 0 underruns
     8 output errors, 2 collisions, 3 interface resets
     0 unknown protocol drops
     0 babbles, 0 late collision, 0 deferred
     1 lost carrier, 0 no carrier, 0 pause output
     0 output buffer failures, 0 output buffers swapped out
```

There are a number of things we can check with the **show interfaces** command. The **input queue drops** indicates that our device received more traffic than it could handle. It's possible that this happens during a traffic peak but it's also possible that the CPU was too busy to handle all the traffic. If you see a lot of drops you might want to investigate further what is going on.

Output queue drops means that you have congestion on the interface. When you receive traffic on a 100Mbit interface and forward it on a 10Mbit interface you'll see congestion which causes packet loss and high delay. Applications that use TCP can use retransmissions but Voice over IP is an application that is very sensitive to packet loss, high delay and a variation in delay which causes jitter. If you see a lot of output queue drops you'll have to start thinking about implementing QoS (Quality of Service).

If you see **input errors** you probably have hardware-related issues. In my example you see 6 CRC errors which indicates that we didn't receive the frames properly. This can be caused because of hardware errors, damaged cabling or duplex mismatches.

Output errors indicate that there was an error with the transmission of the frame. Nowadays we use full-duplex (FastEthernet, Gigabit) so we don't have collisions anymore and the CSMA/CD (Carrier Sense Multi Access / Collision Detection) is disabled.
On half-duplex networks we can have collisions so it's possible to have issues with the transmission of frames. If you do see output errors on a full-duplex network you probably have a duplex mismatch error.

When you are looking at the interface statistics you don't have to panic right away when you see input or output errors. These statistics are logged from the moment the router was booted so it's possible that those errors happened during weeks or months. In my example you can see that we have 299904796 packets that were received on the interface and only 10 input errors…this is no reason for concern. If I would see 100 packets and 20 of them had errors then I would have a reason to further investigate this. It's a good idea to use the **clear counters** command…let it run for a couple of hours or days and then check the

statistics again to get a better picture.

These are all the show command parameters that I wanted to show you for now. If these are new to you I would highly recommend trying them yourself on your Cisco devices! Show commands are useful but they only produce "static" information. If we want to see what is going on real-time on our Cisco devices we'll have to use some debug commands.

*Debugging is very powerful but will put a load on your CPU. Some debug commands produce more output than others. Typing in 'debug ip packet' without a filter will show the output of **ALL** IP packets on your screen…not a good idea!*

You have probably seen a debug before but did you know you can combine them with access-lists for more specific information? This will help you to reduce the load on the CPU and gives you more specific information. Let me give you an example.

```
Router(config)#access-list 100 permit tcp 192.168.12.0 0.0.0.255 any
```

First I will create an access-list that matches on TCP traffic from source network 192.168.12.0 /24 to any destination.

```
Router#debug ip packet 100 detail
IP packet debugging is on (detailed) for access list 100
```

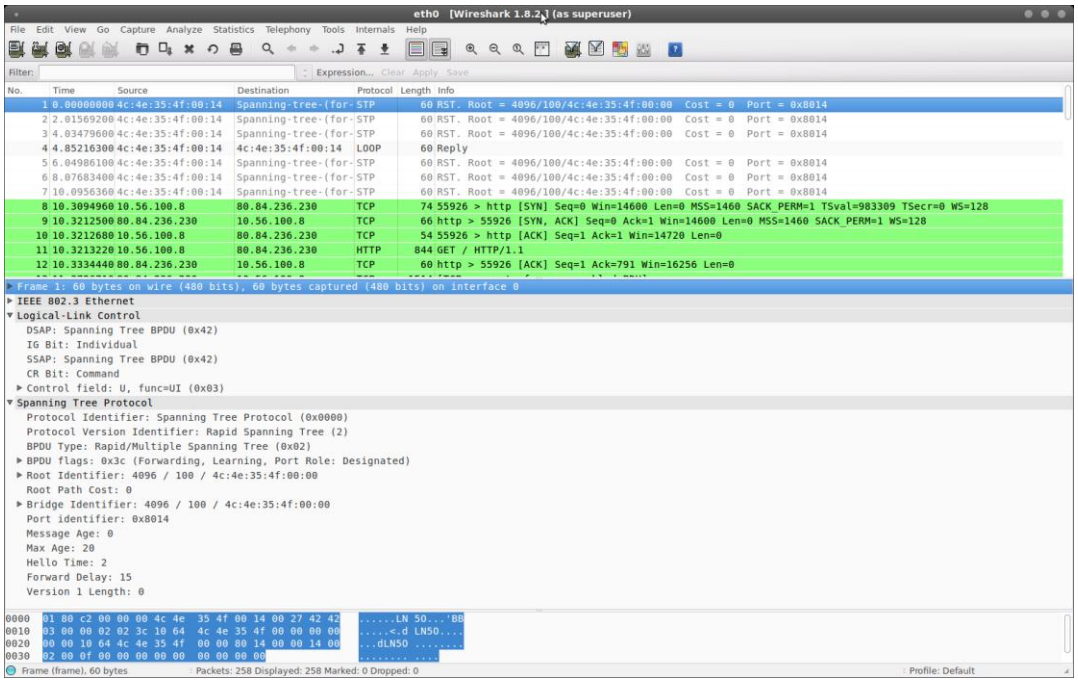
Next step is to enable debugging of IP packets showing details information. I will refer to access-list 100 that I created earlier and use the **detail** parameter to see more specific information.

```
Router#
*Mar  1 03:12:01.127: IP: tableid=0, s=192.168.12.1 (FastEthernet0/0),
d=192.168.12.2 (FastEthernet0/0), routed via RIB
*Mar  1 03:12:01.127: IP: s=192.168.12.1 (FastEthernet0/0), d=192.168.12.2
(FastEthernet0/0), len 44, rcvd 3
*Mar  1 03:12:01.127:     TCP src=40054, dst=23, seq=402773320, ack=0,
win=4128 SYN
*Mar  1 03:12:01.131: IP: tableid=0, s=192.168.12.2 (local), d=192.168.12.1
(FastEthernet0/0), routed via FIB
*Mar  1 03:12:01.135: IP: s=192.168.12.2 (local), d=192.168.12.1
(FastEthernet0/0), len 44, sending
*Mar  1 03:12:01.135:     TCP src=23, dst=40054, seq=2538821922,
ack=402773321, win=4128 ACK SYN
*Mar  1 03:12:01.151: IP: tableid=0, s=192.168.12.1 (FastEthernet0/0),
d=192.168.12.2 (FastEthernet0/0), routed via RIB
*Mar  1 03:12:01.151: IP: s=192.168.12.1 (FastEthernet0/0), d=192.168.12.2
(FastEthernet0/0), len 40, rcvd 3
*Mar  1 03:12:01.151:     TCP src=40054, dst=23, seq=402773321,
ack=2538821923, win=4128 ACK
```

Here's an example of a debug that shows the TCP 3-way handshake. You can see the port number (23) that tells me that this is a session with a telnet server. If you have issues with dropping connections or sessions that won't establish you use a debug like this to see what is going on.

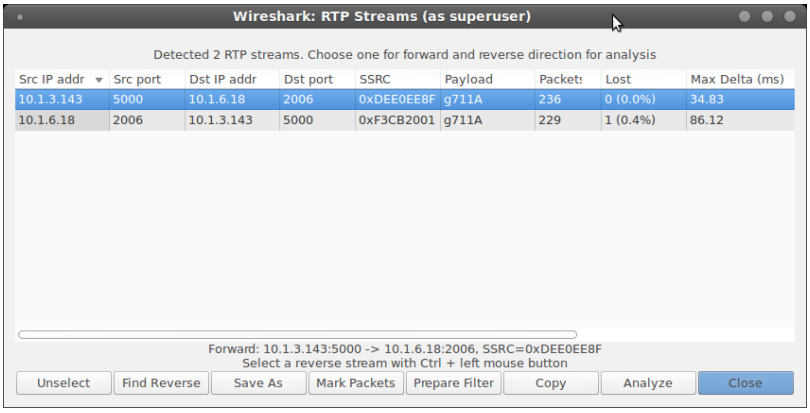
That's all I wanted to show you about debugging for the moment.

Let's continue with some other useful tools.



You have probably seen Wireshark before, it's the most popular protocol analyzer and it's open source. A protocol analyzer is a very powerful tool because you can see the actual traffic on your network. You can spot protocol errors or retransmissions and exactly see what is going on. In order to make this a useful tool there are two things you need to understand:

- Learn how to work with filters so you don't just see "everything" but maybe just the conversation between 2 hosts or perhaps you only want to see the "HTTP" traffic that is going on. You also need to understand how to work with the different graphical views.
- Understand all the different protocols and applications. If I'm having issues with TCP…I need to understand how TCP works…the 3-way handshake, the window size, retransmissions, etc. If you are troubleshooting HTTP related errors you need to understand how HTTP behaves. If you want to solve Voice over IP related issues you need to understand how protocols like SIP work and how VoIP packets are sent using RTP.



To give you an example of the power of Wireshark look at the example above. Voice over IP engineers can use a protocol analyzer like Wireshark to capture all the RTP packets on the network (RTP is used for sending voice samples on the network). Wireshark will filter out the RTP streams for me and it will even show me the delay between the different RTP packets, the packet loss in % and it's even possible to playback the voice conversation!

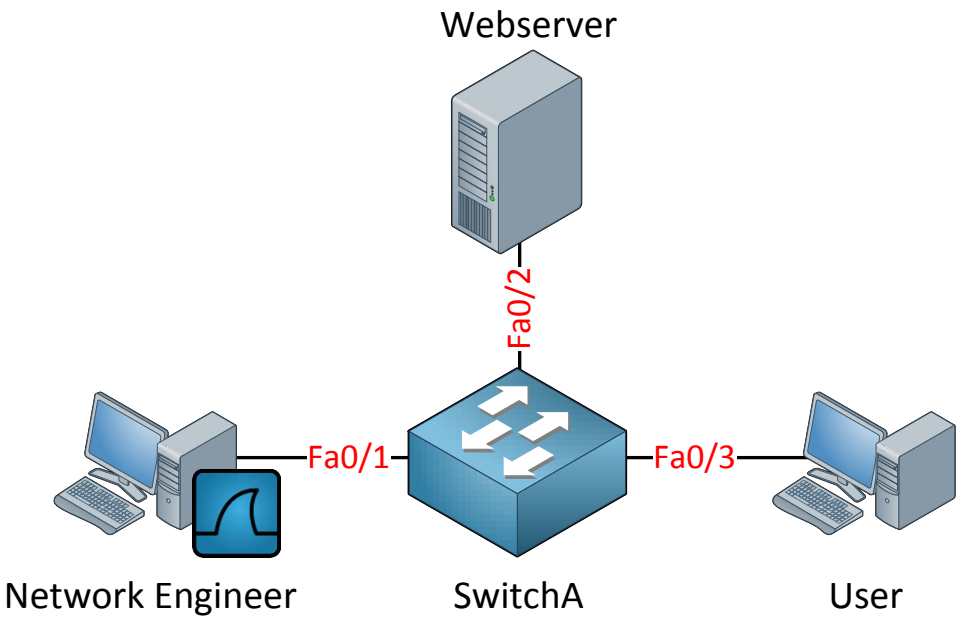
If you want some more examples of Wireshark I recommend you to take a look at the Wireshark website and especially the "Video and Presentations" part. There are some videos where they show you how network issues were solved using Wireshark:

http://www.wireshark.org/docs/

Learning how to work with Wireshark (or any other protocol analyzer) isn't something you learn in 5 minutes but it's well worth your time. If you want to become an expert at Wireshark I can highly recommend you to take a look at the Wireshark Network Analysis book written by Laura Chappell:

http://www.wiresharkbook.com/

If you want to capture traffic and analyze it with a protocol analyzer there are a number of things we have to think about.

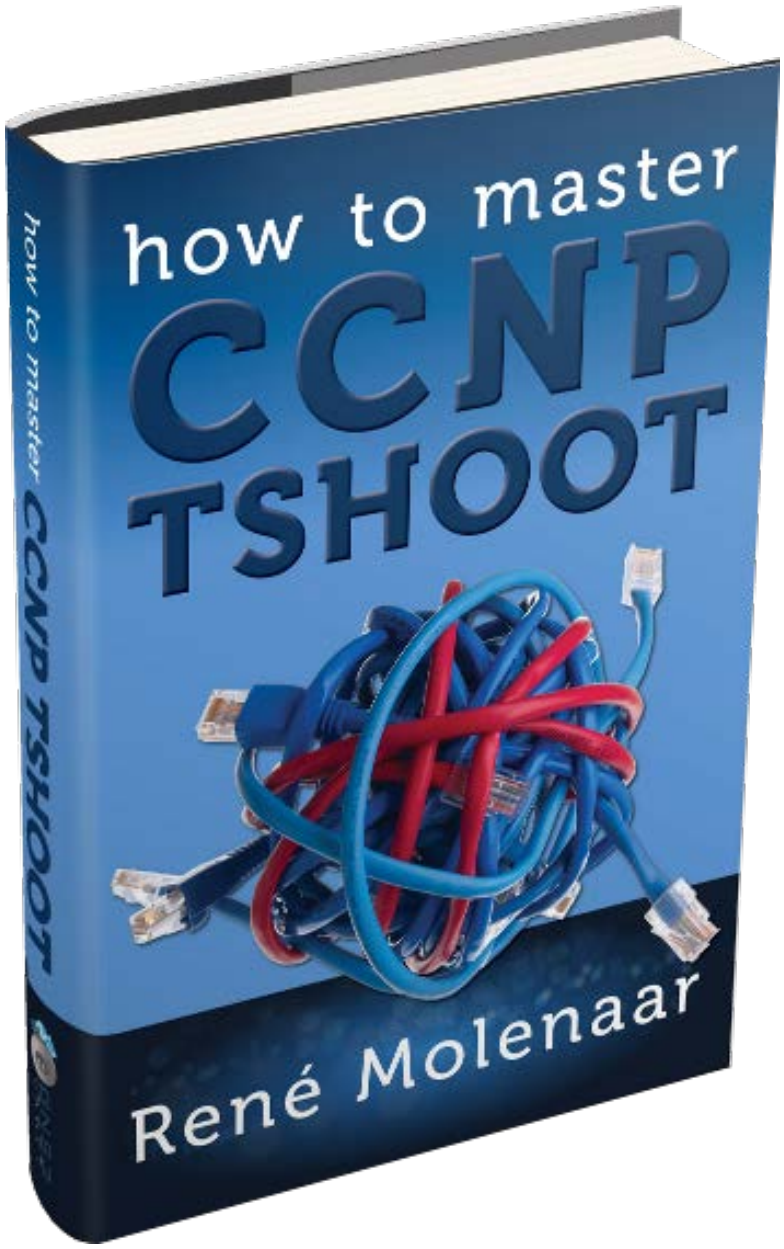


Take a look at the picture above. Our network engineer is running Wireshark on his computer and is connected to the fa0/1 interface of SwitchA. A user is connected to fa0/3 and there's a webserver connected to the fa0/2 interface. The user is browsing some webpage from the webserver and is having issues that our network engineer wants to investigate. As soon as we start Wireshark the only traffic that you will capture is **between your computer and the switch**. That's right you'll only see whatever is happening on the fa0/1 interface. If we want to see the traffic between our user and the webserver we'll have to configure **SPAN**.

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

Click on the link below to get the full version.

Get How to Master CCNP TSHOOT Today



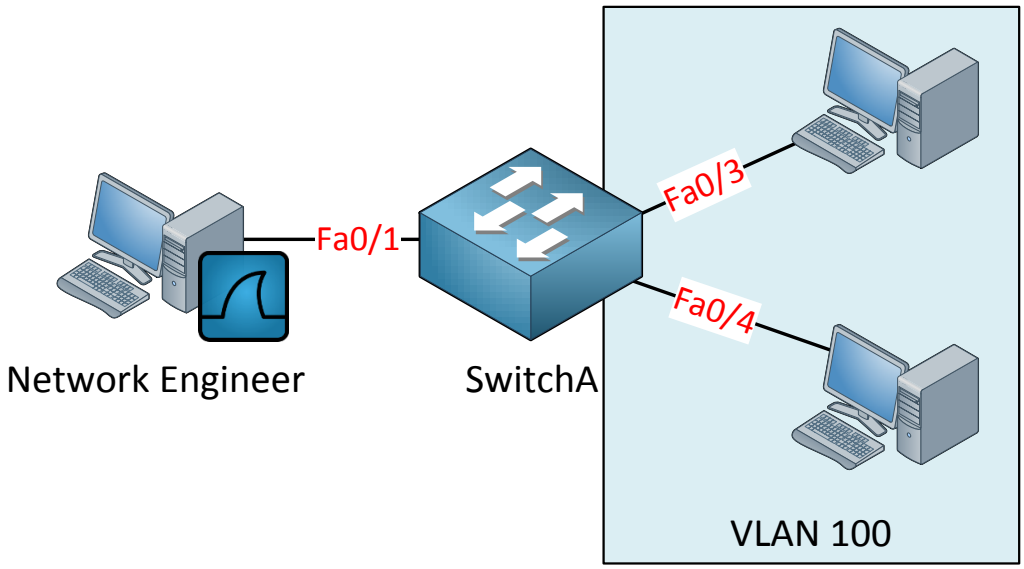
SPAN (Switched Port Analyzer) will copy traffic from one interface or VLAN to an interface. Let's take a look how we can use SPAN to see traffic from the webserver:

```
SwitchA(config)#monitor session 1 source interface fa0/2
SwitchA(config)#monitor session 1 destination interface fa0/1
```

We use the **monitor session** command to configure SPAN. You can pick any session number you like…I used number 1. In my example I'm copying all traffic from the fa0/2 interface (the webserver) to the fa0/1 interface (our network engineer's computer). If you start sniffing with Wireshark you'll see everything that happens on the fa0/2 interface.

```
SwitchA#show monitor
Session 1
---------
Type                   : Local Session
Source Ports           :
    Both               : Fa0/2
Destination Ports      : Fa0/1
    Encapsulation      : Native
          Ingress      : Disabled
```

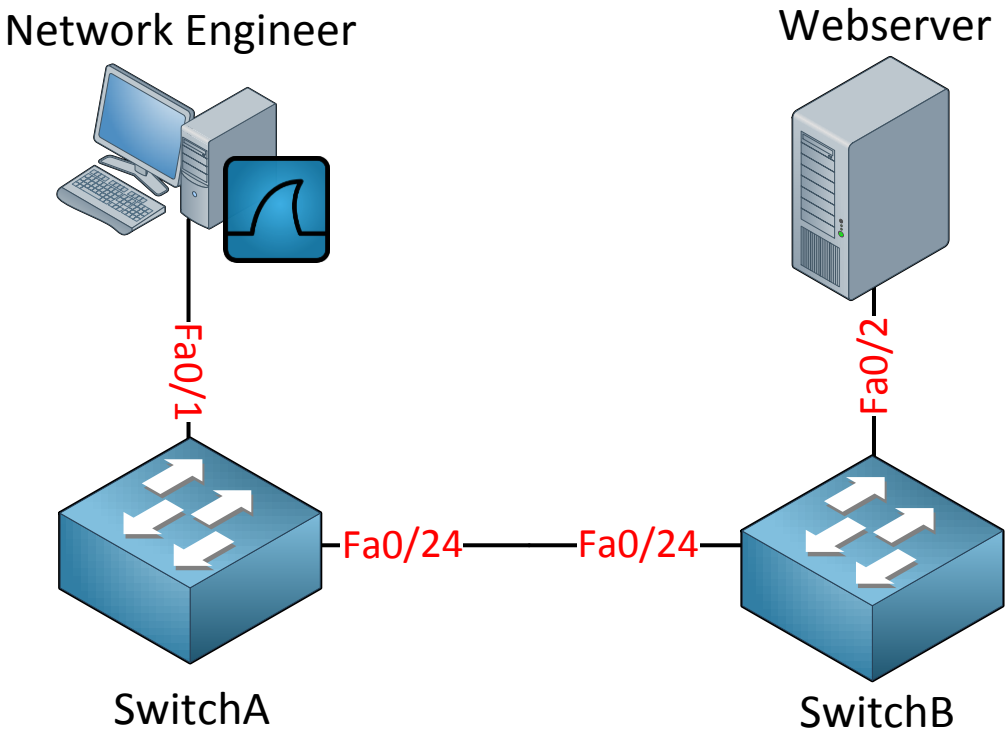
You can use the **show monitor** command to verify your configuration.



We can also use SPAN to copy traffic from entire VLAN to a destination interface. In the example above we have 2 computers in VLAN 100, let's copy their traffic to the fa0/1 interface so we can sniff it.

```
SwitchA(config)#monitor session 1 source vlan 100
SwitchA(config)#monitor session 1 destination interface fa0/1
```

Configure SPAN to use VLAN 100 as the source and copy it to the fa0/1 interface. All traffic from VLAN 100 will now be copied so we can sniff it!



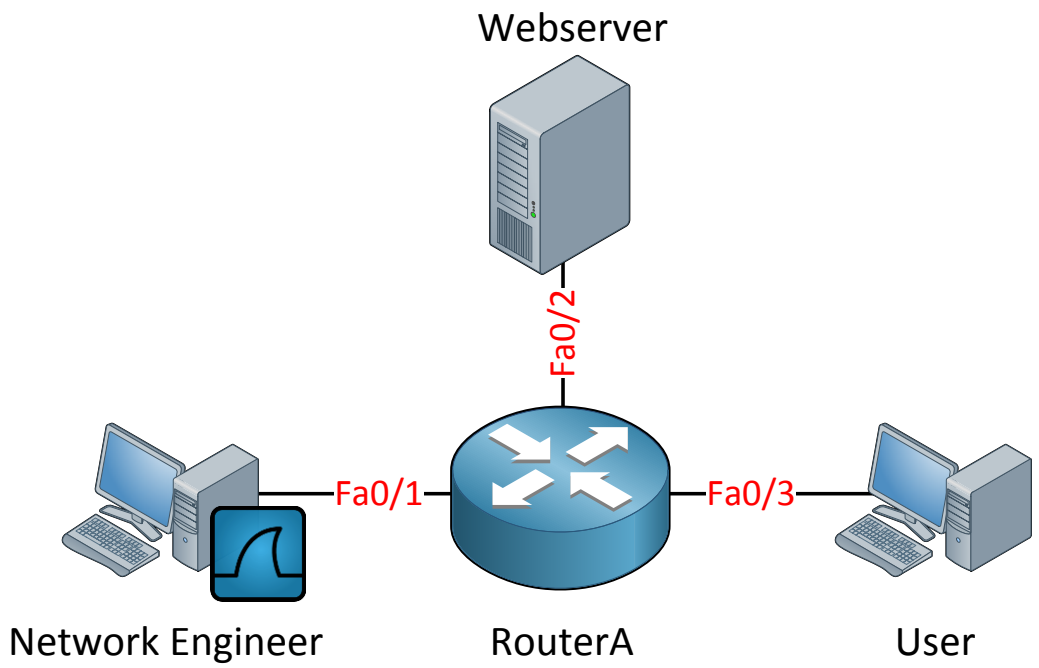
What if the webserver is connected to another switch? This time the webserver is connected to SwitchB while our network engineer is still connected to SwitchA. **RSPAN (Remote SPAN)** to the rescue! It's possible to sniff traffic from a remote switch, let me show you how:

```
SwitchB(config)#vlan 50
SwitchB(config-vlan)#name RSPAN
SwitchB(config-vlan)#remote-span
SwitchB(config-vlan)#exit
SwitchB(config)#monitor session 1 source interface fa0/2
SwitchB(config)#monitor session 1 destination remote vlan 50
```

Since we are sending the traffic from SwitchB to SwitchA we'll need something to carry our traffic. We have to use a VLAN for this and tell the switch that we'll use it for remote SPAN. In my example I'm using VLAN 50. Traffic from the fa0/2 interface on SwitchB should be copied to VLAN 50.

```
SwitchA(config)#vlan 50
SwitchA(config-vlan)#name RSPAN
SwitchA(config-vlan)#remote-span
SwitchA(config-vlan)#exit
SwitchA(config)#monitor session 1 destination interface fa0/1
SwitchA(config)#monitor session 1 source remote vlan 50
```

On SwitchA we'll configure RSPAN so everything from VLAN 50 will be copied to the fa0/1 interface. This will ensure that we can sniff traffic from the webserver. SPAN and RSPAN are very useful but depending on the switch platform you are using it might have some limitations, it's best to check the configuration guide of your switch to find out what options you can use.



What if we have a router instead of a switch? SPAN doesn't exist for routers but we do have something called **RITE (Router IP Traffic Export).** RITE allows us to copy IP packets to a destination interface just like SPAN does.
Let me show you an example:

```
RouterA(config)#ip traffic-export profile MYEXPORT
RouterA(conf-rite)#interface fastEthernet 0/1
RouterA(conf-rite)#mac-address 1111.1111.1111
```

This is RITE in its most simple form. We use the **ip traffic-export profile** command to create a new profile which I called "MYEXPORT". Next step is to specify the destination interface which will be fa0/1 in my case. We also have to configure the MAC address of the device that is running Wireshark.

```
RouterA(conf-rite)#bidirectional
```

By default only **inbound** IP packets will be copied to the destination interface. Use the **bidirectional** command to copy inbound AND outbound IP packets.

```
RouterA(conf-rite)#incoming access-list ?
  <1-199>       IP access list (standard or extended)
  <1300-2699>   IP expanded access list (standard or extended)
  WORD          Access-list name
```

Optionally you can apply an access-list. In my example I'm only interested in traffic from the webserver so I could use an access-list that will only match traffic on destination port 80 (HTTP) and port 443 (HTTPS).

```
RouterA(config)#interface fa0/2
RouterA(config-if)#ip traffic-export apply MYEXPORT
%RITE-5-ACTIVATE: Activated IP traffic export on interface FastEthernet0/2
```

We still need to activate RITE on the interface. In my example I want to copy all IP packets from the webserver so I'll type **ip traffic-export apply MYEXPORT** on the fa0/2 interface that is connected to the webserver. You'll see a notification message that RITE has been activated. All IP packets from the webserver will now be copied to the fa0/1 interface!

That's all that I have for you about Wireshark, SPAN and RITE. Let's continue by looking at ping and telnet.

Let's start with the **ping** command. You are probably familiar with the ping command to see if you can reach certain IP addresses but this command is a classic example of *"there's more than meets the eye".*

```
Router#ping 192.168.12.2 repeat 1000
```

The first parameter **"repeat"** for ping is easy. By default 5 IP packets will be sent, in my example above I'm sending 1000. This can be useful if you have packet loss in your network, you can use the ping command as a quick tool to generate traffic.

```
Router#ping 192.168.12.2 size 1500
```

We can change the size of our IP packet with the **"size"** parameter. This is useful for 2 reasons:

- You can create larger packets to see what the maximum MTU (maximum transmission unit) is in the network.
- You can combine it with the repeat parameter to generate some load on your network.

```
Router#ping 192.168.12.2 size 1476 df-bit
Type escape sequence to abort.
Sending 5, 1476-byte ICMP Echos to 192.168.12.2, timeout is 2 seconds:
Packet sent with the DF bit set
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 184/189/193 ms
```

If you want to check the maximum MTU you need to use the **df-bit (don't fragment)** parameter. In the example above I have set the size of the IP packet to 1476 bytes and you can see that the ping is working.

```
Router#ping 192.168.12.2 size 1477 df-bit
Type escape sequence to abort.
Sending 5, 1477-byte ICMP Echos to 10.1.221.1, timeout is 2 seconds:
Packet sent with the DF bit set
M.M.M
Success rate is 0 percent (0/5)
```

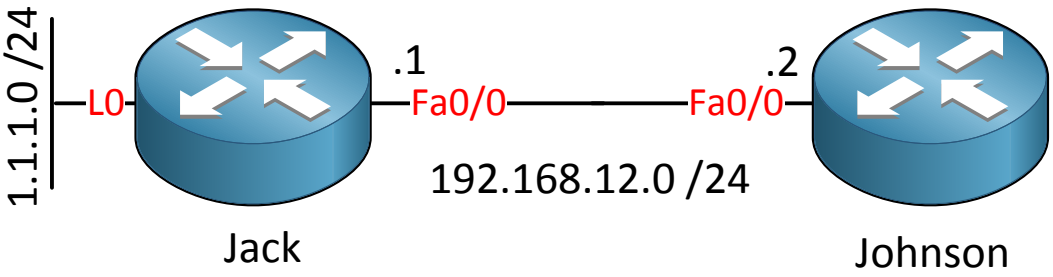
In this example I've set the size to 1477 bytes and you can see the pings are failing. This can sometimes occur when you are using a tunneling mechanism like GRE or IPSEC.

The tunnel will add overhead so it's possible that we exceed the maximum MTU of 1500 bytes. We can conclude that there is a host in the path to destination 192.168.12.2 that has a maximum MTU size of 1476 bytes.

In the previous example I have to specify the size myself which is kinda annoying, it's also possible to use a variety of IP packet sizes so we don't have to do the guesswork ourselves.

```
Router#ping
Protocol [ip]:
Target IP address: 192.168.12.2
Repeat count [5]: 1
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:
Type of service [0]:
Set DF bit in IP header? [no]: yes
Validate reply data? [no]:
Data pattern [0xABCD]:
www.CareerCert.info
Loose, Strict, Record, Time stamp, Verbose[none]:
Sweep range of sizes [n]: y
Sweep min size [36]: 1300
Sweep max size [18024]: 1500
Sweep interval [1]:
Type escape sequence to abort.
Sending 101, [1300..1500]-byte ICMP Echos to 10.1.221.1, timeout is 2
seconds:
Packet sent with the DF bit set
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!M.M.M.M.M.M.M.M.M.M.M.
Success rate is 82 percent (83/101), round-trip min/avg/max = 4/4/16 ms
```

Use the ping command without any parameter and just hit enter. It will ask you for a variety of options. In this case I want to set a size of ranges. In the example above I set the lowest MTU size to 1300 and the highest MTU size to 1500. The first pings are OK and the ones with the larger MTU size fail. 83 IP packets were sent successfully and we started with a MTU size of 1300 bytes. This means there's a host that has a maximum MTU size of 1383 bytes.



The ping command can also be used to check routing issues by using the **source** parameter. In the example above I have 2 routers, they are directly connected using their Fa0/0 interfaces and they have an IP address from the 192.168.12.0 /24 subnet. Router Jack also has a

loopback0 interface with IP address 1.1.1.1 /24.

```
Jack#ping 192.168.12.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/8/24 ms
```

If I use the ping command my router will use the IP address of the directly connected interface, this is what the IP packet will look like:

- Source: 192.168.12.1
- Destination: 192.168.12.2

Router Johnson will receive the IP packet, determine the packet is meant for him and will respond to the ping (ICMP echo-request) with an ICMP echo-reply. The packet will look like this:

- Source: 192.168.12.2
- Destination: 192.168.12.1

Router Johnson will do a routing table lookup and determines that IP address 192.168.12.1 falls within network 192.168.12.0 /24 which is directly connected for him. The IP packet can be sent towards router Jack.

```
Jack#ping 192.168.12.2 source loopback 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.2, timeout is 2 seconds:
Packet sent with a source address of 1.1.1.1
.....
Success rate is 0 percent (0/5)
```

This time I'm sending another IP packet with an ICMP echo-request but we change the source IP address to 1.1.1.1 (loopback0) using the **source** parameter. The IP packet will look like this:

- Source: 1.1.1.1
- Destination: 192.168.12.2

The IP packet will make it to router Johnson; he will respond with an ICMP echo-reply and do another routing table lookup. This time we need to figure out where to send IP packets with destination address 1.1.1.1. Router Johnson doesn't have anything in its routing table that matches this IP address and will drop the packet. As a result my pings are failing. By using the ping command like this we can check if a router has certain entries in its routing table or not.

The next tool I want to talk about is telnet. Telnet uses TCP and port 23 but we can also use it to connect to other ports.

```
Router#telnet 192.168.12.2 80
Trying 192.168.12.1, 80 ... Open
123test blablabla
HTTP/1.1 400 Bad Request
Date: Fri, 01 Mar 2002 02:36:35 GMT
Server: cisco-IOS
Accept-Ranges: none

400 Bad Request

[Connection to 192.168.12.2 closed by foreign host]
```

I can telnet to other port numbers. If I see a message that says **open** like my example above I know that I can connect to this port number using TCP. Some applications might even respond to my input.

```
Router#telnet 192.168.12.1 22
Trying 192.168.12.1, 22 ... Open
SSH-1.99-Cisco-1.25


[Connection to 192.168.12.1 closed by foreign host]
```

Here's another example but this time I telnet to port 22 (SSH). You can see the SSH server shows me a banner.

```
Router#telnet 192.168.12.1 22 /source-interface loopback 0
```

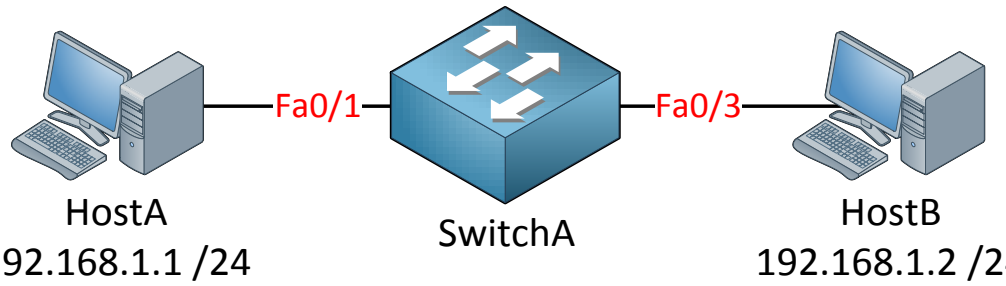
Besides changing the port number to connect to we can also specify the source interface. This will change the source IP address. This can be useful if you want to test certain access-lists and see if your IP packets match your access-list statements where you filter on TCP traffic.

These are all the tools that I wanted to share with you and this is the end of the chapter. Hopefully you learned some new tricks that you can add to your troubleshooting toolkit.

Now it's finally time to dive into troubleshooting some protocols!

3. Troubleshooting Switching

If we want to work our way from the bottom of the OSI model to the top we'll have to start with the protocols that we use on the switches. Think about VLANs, trunking, etherchannels and spanning-tree. I'm going to walk you through the different protocols and I'll show you different scenarios where "something" is wrong. We'll tackle these problems by using a combination of show and debug commands. First stop…interface issues!



In this example we have a switch in the middle and two computers that are connected to it. Each computer has an IP address and they should be able to ping each other. We'll assume the computers are configured correctly and there are no issues there (after all this isn't a Windows / MAC / Linux course).

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Unfortunately our pings are not working. What's the first thing we should check? Our interfaces of course!

```
SwitchA#show interfaces fa0/1
FastEthernet0/1 is down, line protocol is down (notconnect)
  Hardware is Fast Ethernet, address is 0011.bb0b.3603 (bia 0011.bb0b.3603)
  MTU 1900 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Half-duplex, Auto-speed, media type is 10/100BaseTX
  input flow-control is off, output flow-control is unsupported
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:26:47, output 00:19:17, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
```

```
     3457 packets input, 309301 bytes, 0 no buffer
     Received 2407 broadcasts (1702 multicasts)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog, 1702 multicast, 0 pause input
     0 input packets with dribble condition detected
     42700 packets output, 8267872 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier, 0 PAUSE output
     0 output buffer failures, 0 output buffers swapped out
```

FastEthernet 0/1 is showing down. This could indicate a layer 1 problem like a broken cable, wrong cable (crossover instead of straight-through) or maybe a bad NIC. Note that this interface is running in **half duplex**. If you are lucky you might get a duplex message through CDP that tells you that there is a duplex mismatch. If you are unlucky it's possible that your interface goes down. Keep in mind that a Gigabit interface doesn't support half-duplex.

```
SwitchA(config)#interface fa0/1
SwitchA(config-if)#duplex auto
```

I'll change the interface to duplex auto so the switch can figure it out by itself.

```
SwitchA#
%LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed
state to up
```

That's looking better!

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Maybe we are lucky…not this time, the ping isn't working.

```
SwitchA#show interfaces fa0/3
FastEthernet0/3 is down, line protocol is down (notconnect)
  Hardware is Fast Ethernet, address is 0011.bb0b.3605 (bia 0011.bb0b.3605)
  MTU 1900 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Auto-duplex, 10Mb/s, media type is 10/100BaseTX
  input flow-control is off, output flow-control is unsupported
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:38:09, output 00:01:42, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     1908 packets input, 181819 bytes, 0 no buffer
     Received 858 broadcasts (826 multicasts)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog, 826 multicast, 0 pause input
     0 input packets with dribble condition detected
     46861 packets output, 9365341 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier, 0 PAUSE output
     0 output buffer failures, 0 output buffers swapped out
```

Interface fa0/3 that is connected to HostB is also down. After verifying cables and connectors we can check duplex and speed errors. Duplex is on auto so that shouldn't be a problem. Speed has been set to 10Mbit however while this interface is a FastEthernet (100Mbit) link.

```
SwitchA(config)#interface fa0/3
SwitchA(config-if)#speed auto
```

Let's change the speed to auto and see what happens.

```
SwitchA#
%LINK-3-UPDOWN: Interface FastEthernet0/3, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/3, changed
state to up
```

It seems the speed mismatch caused the interface to go down. Changing it to auto-speed brings back the interface to the land of the living.

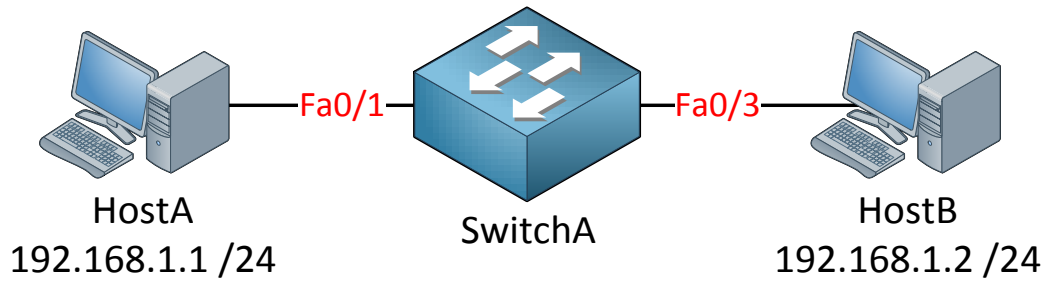
```
SwitchA#show ip interface brief
Interface              IP-Address      OK? Method Status
Protocol
FastEthernet0/1        unassigned      YES unset  up                    up
FastEthernet0/3        unassigned      YES unset  up                    up
```

This is what we are looking for. The interfaces that I'm working with are both showing

up/up. At least we now know that there are no cable, speed or duplex errors.

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Now our ping is working. **Lesson learned: Check your interfaces and see if they show as up/up.**



Same topology but there's a different problem here.

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Request timed out.
Request timed out.

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

HostA is unable to ping HostB. We'll start by checking the interfaces:

```
SwitchA#show ip interface brief
Interface              IP-Address      OK? Method Status
Protocol
FastEthernet0/1        unassigned      YES unset  down                  down
FastEthernet0/3        unassigned      YES unset  up                    up
```

FastEthernet 0/3 is looking fine but something is wrong with FastEthernet 0/1.

Let's take a closer look at it:

```
SwitchA#show interfaces fa0/1
FastEthernet0/1 is down, line protocol is down (err-disabled)
```

Hmm it says **err-disabled**. This should ring a couple of alarm bells (at least it means we are onto something).

```
SwitchA#show interfaces status err-disabled

Port  Name               Status       Reason               Err-disabled
Vlans
Fa0/1                    err-disabled psecure-violation
```

Use the **show interfaces status err-disabled** command to see **why** the interface got into error-disabled mode. It's telling me port-security is the reason.

```
SwitchA#show port-security interface fa0/1
Port Security              : Disabled
Port Status                : Secure-shutdown
Violation Mode             : Shutdown
Aging Time                 : 0 mins
Aging Type                 : Absolute
SecureStatic Address Aging : Disabled
Maximum MAC Addresses      : 1
Total MAC Addresses        : 1
Configured MAC Addresses   : 1
Sticky MAC Addresses       : 0
Last Source Address:Vlan   : 000c.2928.5c6c:1
Security Violation Count   : 1
```

We can look at the port security configuration and we see that only 1 MAC address is allowed. The last MAC address seen on the interface is 000c.2928.5c6c.

```
SwitchA#show port-security interface fa0/1 address
          Secure Mac Address Table
------------------------------------------------------------------------
Vlan    Mac Address       Type                      Ports   Remaining Age
                                                              (mins)
----    -----------       ----                      -----   -------------
   1    0019.569d.5742    SecureConfigured          Fa0/1         -
------------------------------------------------------------------------
Total Addresses: 1
```

Here we see that another MAC address has been configured for port security. This is the reason that the port went into err-disabled mode.

```
SwitchA(config)#interface fa0/1
SwitchA(config-if)#no switchport port-security
```

Let's get rid of port security to fix the problem.

```
SwitchA(config)#interface fa0/1
SwitchA(config-if)#shutdown
SwitchA(config-if)#no shutdown
```

This is something you should not forget. After getting rid of the port security configuration your interface is **still** in err-disabled mode. You need to do a shutdown and no shutdown to make it alive again!

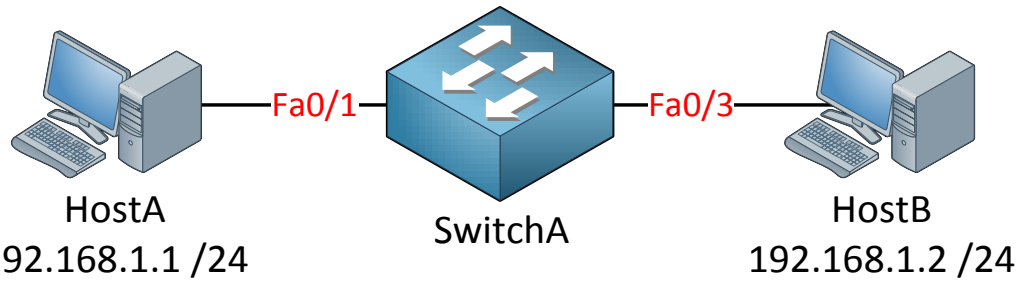
```
SwitchA#
%LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed
state to up
```

The console tells us that the interface is now up.

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

We can now ping between the computers. Problem solved**! Lesson learned: Check if an interface is in err-disabled and if so: A) check why this happened and B) solve the problem.**

*Not seeing err-disabled doesn't automatically mean there are **no** port-security issues. The default **violation mode** for port security is **shutdown** which will put the interface in **err-disabled** mode. The **restrict** mode will keep the interface up but shows a log message on the console. **Protect** mode also keeps the interface up but **doesn't** show any console messages. It's not a bad idea to take a **quick look** to see if port security is active or not…it's also a good idea to use **show mac address-table** to see if the switch learned the MAC addresses on the interfaces.*



Let's continue with another problem. Same topology but something else is wrong.

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

The two computers are unable to ping each other (what a surprise!).

```
SwitchA#show ip int brief
Interface              IP-Address      OK? Method Status
Protocol
FastEthernet0/1        unassigned      YES unset  up                    up
FastEthernet0/3        unassigned      YES unset  up                    up
```

The interfaces are looking good, no errors here.

```
SwitchA#show port-security
Secure Port  MaxSecureAddr  CurrentAddr  SecurityViolation  Security Action
                (Count)       (Count)          (Count)
---------------------------------------------------------------------------
---------------------------------------------------------------------------
Total Addresses in System (excluding one mac per port)     : 0
Max Addresses limit in System (excluding one mac per port) : 6144
```

Let's practice what I preach. Port security is disabled on this switch as you can see above. At this moment we at least know that there are no interface issues and port security isn't filtering any MAC addresses.

```
SwitchA#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
----
1    default                          active    Fa0/1, Fa0/2, Fa0/4, Fa0/5
                                                Fa0/6, Fa0/7, Fa0/8, Fa0/9
                                                Fa0/10, Fa0/11, Fa0/12,Fa0/13
                                                Fa0/14, Fa0/15, Fa0/16,Fa0/17
                                                Fa0/18, Fa0/19, Fa0/20,Fa0/21
                                                Fa0/22, Fa0/23, Fa0/24, Gi0/1
                                                Gi0/2
2    VLAN0002                         active    Fa0/3
```

At this moment it's a good idea to check the VLAN information. You can use the **show vlan** command to quickly verify to which VLAN the interfaces belong.

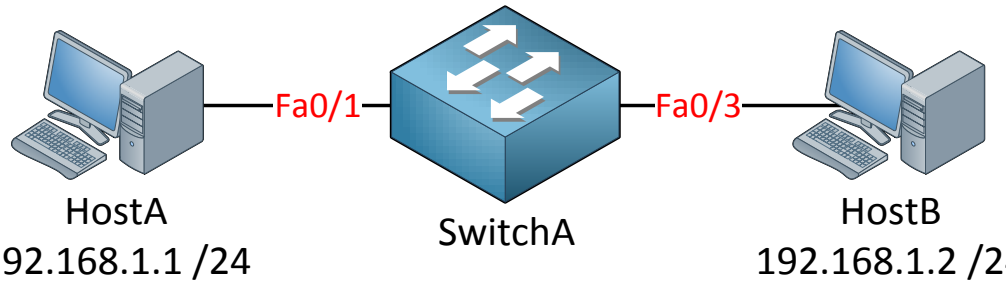
As you can see our interfaces are not in the same VLAN.

```
SwitchA(config)#interface fa0/3
SwitchA(config-if)#switchport access vlan 1
```

We'll move interface fa0/3 back to VLAN 1.

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Both computers are now in the same VLAN. This solves our problem! **Lesson learned: Make sure the interface is in the correct VLAN.**



Time for another problem! Our two computers are unable to ping each other and I think by now you know what a failed ping looks like so I won't post it again.

```
SwitchA#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/1        unassigned      YES unset  up                    up
FastEthernet0/3        unassigned      YES unset  up                    up
```

The interfaces don't show any errors.

```
SwitchA#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------
----
1    default                           active    Fa0/2, Fa0/4, Fa0/5, Fa0/6
                                                 Fa0/7, Fa0/8, Fa0/9, Fa0/10
                                                 Fa0/11, Fa0/12,
Fa0/13,Fa0/14
                                                 Fa0/15, Fa0/16,
Fa0/17,Fa0/18
                                                 Fa0/19, Fa0/20,
Fa0/21,Fa0/22
                                                 Fa0/23, Fa0/24, Gi0/1,
Gi0/2
10   VLAN0010                          active    Fa0/1
```

We'll take a look at the VLAN assignment. You can see that FastEthernet 0/1 is in VLAN 10 but I don't see FastEthernet 0/3 anywhere. Here are the possible causes:

- Something is wrong with the interface. We proved this wrong because it shows up/up so it seems to be active.
- The interface is not an access port but a **trunk**.

```
SwitchA#show interfaces fa0/3 switchport
Name: Fa0/3
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: On
Access Mode VLAN: 10 (VLAN0010)
Trunking Native Mode VLAN: 1 (default)
```

A quick look at the switchport information shows us what we need to know. We can confirm that interface fa0/3 is in trunk mode and the native VLAN is 1. This means that whenever HostB sends traffic and doesn't use 802.1Q tagging that our traffic ends up in VLAN 1.

```
SwitchA(config)#interface fa0/3
SwitchA(config-if)#switchport mode access
SwitchA(config-if)#switchport access vlan 10
```

We'll turn fa0/3 into access mode and make sure it's in VLAN 10.

```
SwitchA#show vlan id 10

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------
----
10   VLAN0010                          active    Fa0/1, Fa0/3
```

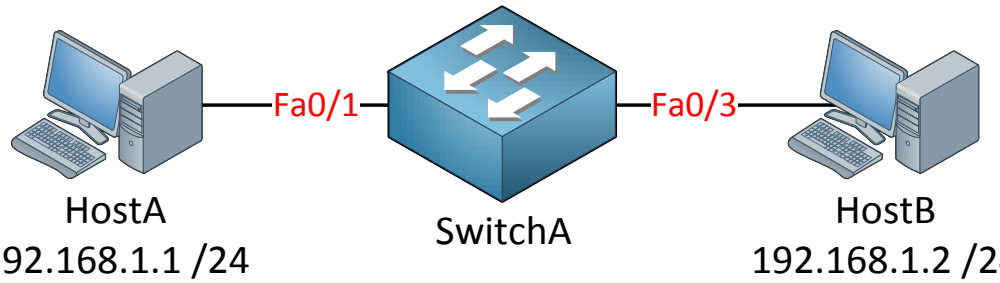
Both interfaces are now active in VLAN 10.

```
SwitchA#show interfaces fa0/3 switchport | include Operational Mode
Operational Mode: static access
```

It's maybe better to check the switchport information.

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Now I can send a ping from HostA to HostB…problem solved! **Lesson learned: Make sure the interface is in the correct switchport mode (access or trunk mode).**



Same two computers, same switch. This scenario is a bit more interesting though. The computers are unable to ping each other so let's walk through our list of "possible" errors:

```
SwitchA#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/1        unassigned      YES unset  up                    up
FastEthernet0/3        unassigned      YES unset  up                    up
```

The interfaces are looking good, up/up is what we like to see.

```
SwitchA#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
----
1    default                           active    Fa0/2, Fa0/4, Fa0/5, Fa0/6
                                                 Fa0/7, Fa0/8, Fa0/9, Fa0/10
                                                 Fa0/11, Fa0/12, Fa0/13,
Fa0/14
                                                 Fa0/15, Fa0/16, Fa0/17,
Fa0/18
                                                 Fa0/19, Fa0/20, Fa0/21,
Fa0/22
                                                 Fa0/23, Fa0/24, Gi0/1, Gi0/2
10   VLAN0010                          active    Fa0/1, Fa0/3
```

Both interfaces are in VLAN 10 so this looks ok to me.

```
SwitchA#show port-security
Secure Port  MaxSecureAddr  CurrentAddr  SecurityViolation  Security Action
                (Count)       (Count)          (Count)
---------------------------------------------------------------------------
---------------------------------------------------------------------------
Total Addresses in System (excluding one mac per port)     : 0
Max Addresses limit in System (excluding one mac per port) : 6144
```

Just to be sure…there's no port security. This is an interesting situation. The interfaces are up/up, we are in the same VLAN and there's no port security. Anything else that could block traffic?

```
SwitchA#show vlan filter
VLAN Map BLOCKSTUFF is filtering VLANs:
  10
```

You bet! This might not be something you think about immediately but we can use VACLs (VLAN access-list) to permit or deny traffic within the VLAN. If you are troubleshooting switches then this is something you'll have to look at if everything else seems fine. In this case there's a VACL attached to VLAN 10, let's inspect it.

```
SwitchA#show vlan access-map
Vlan access-map "BLOCKSTUFF"  10
  Match clauses:
    ip  address: 1
  Action:
    drop
Vlan access-map "BLOCKSTUFF"  20
  Match clauses:
  Action:
    forward
```

There are two sequence numbers…10 and 20. Sequence number 10 matches on access-list 1 and the action is to drop traffic. Let's take a look what this access-list 1 is about:

```
SwitchA#show access-lists
Standard IP access list 1
    10 permit 192.168.1.0, wildcard bits 0.0.0.255
```

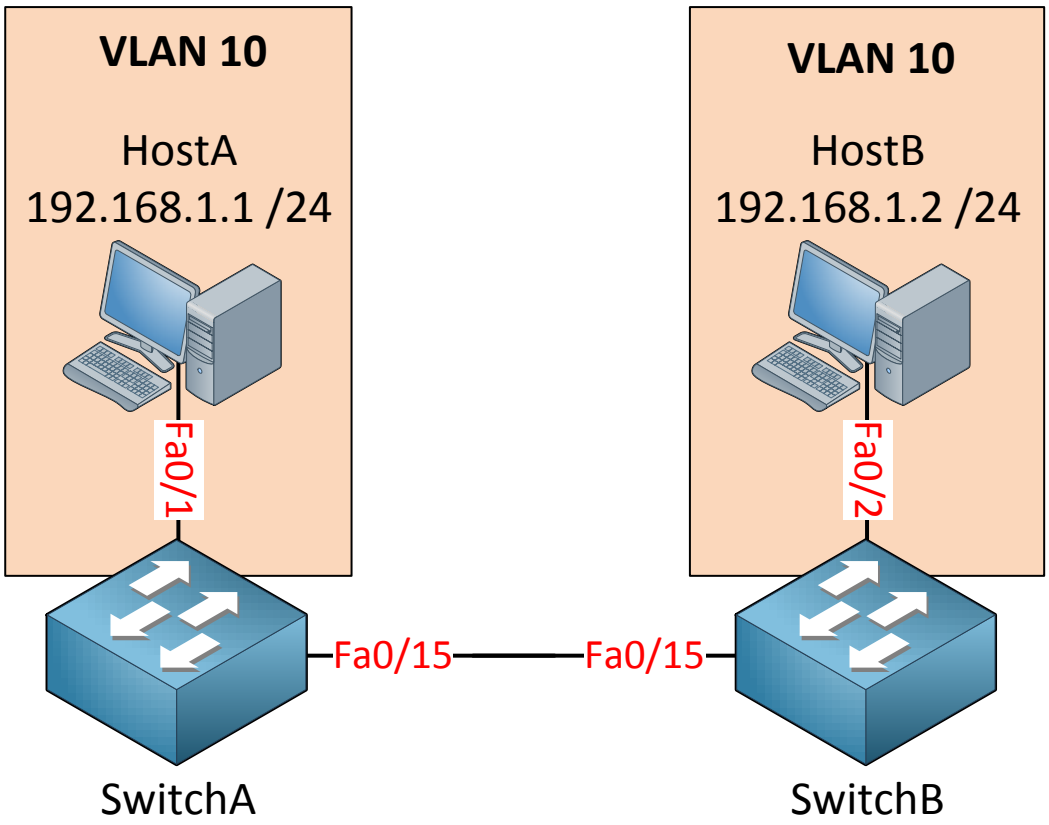
Don't be confused because of the permit statement here. Using a permit statement in the access-list means that it will "match" on 192.168.1.0 /24. Our two computers are using IP addresses from this range. If it matches this access-list then the VLAN access-map will drop the traffic.

```
SwitchA(config)#vlan access-map BLOCKSTUFF 10
SwitchA(config-access-map)#action forward
```

Let's change the action to forward and see if it solves our problem.

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

There we go, our ping is now working. **Lesson learned: If everything else seems to be ok, make sure there's no VACL!**



Let's continue with another topology. By now you know we need to check the interfaces first and then the VLANS. In this example I have the same two computers but now we have two switches. The ping from HostA to HostB is failing so where do we start looking?

```
SwitchA#show interfaces fa0/1
FastEthernet0/1 is up, line protocol is up (connected)
```

```
SwitchA#show interfaces fa0/1 switchport
Name: Fa0/1
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: On
Access Mode VLAN: 10 (VLAN0010)
```

```
SwitchA#show port-security interface fa0/1
Port Security              : Disabled
```

First I'll verify the fa0/1 interface on Switch1. The interface is up and running, it's a switchport and assigned to VLAN 10. This is looking good so far. Port security is not enabled so we don't have to worry about it.

```
SwitchB#show interfaces fa0/2
FastEthernet0/2 is up, line protocol is up (connected)
```

```
SwitchB#show interfaces fa0/2 switchport
Name: Fa0/2
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: On
Access Mode VLAN: 10 (VLAN0010)
```

```
SwitchB#show port-security interface fa0/2
Port Security              : Disabled
```

I'll check the same things on Switch2. The interface is operational and it has been assigned to VLAN 10.

At this moment we know that the interfaces to the computers are looking good. At this moment you could do two things:

- Connect another computer to Switch1 and assign it to VLAN 10. See if you can communicate between computers in VLAN 10 when they are connected to the same switch. Do the same on Switch2.
- Check the interface between Switch1 and Switch2.

I'm going to focus on the interface between Switch1 and Switch2 because there's plenty that could go wrong there!

```
SwitchA#show interfaces fa0/15
FastEthernet0/15 is up, line protocol is up (connected)
```

```
SwitchB#show interfaces fa0/15
FastEthernet0/15 is up, line protocol is up (connected)
```

The interfaces are showing no issues, time to check the switchport information.

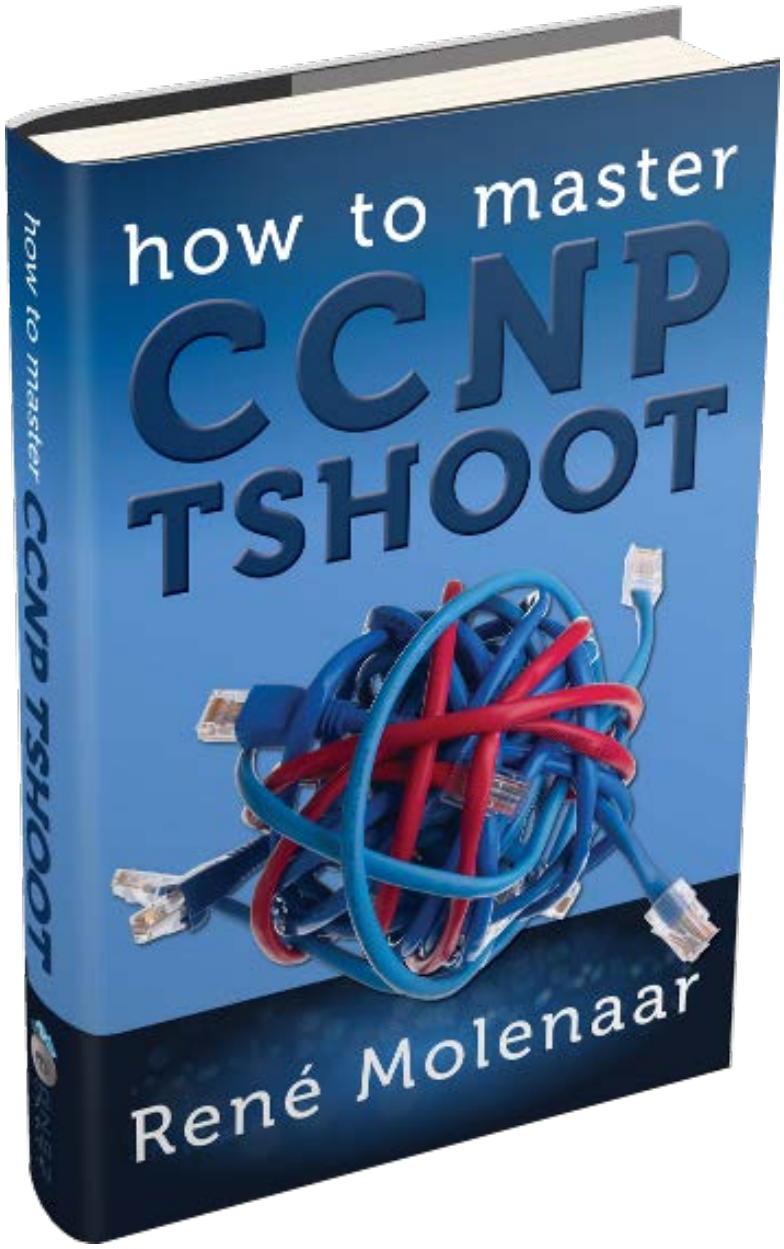
```
SwitchA#show interfaces fa0/15 switchport
Name: Fa0/15
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: isl
Operational Trunking Encapsulation: isl
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
```

SwitchA is in trunk mode and using **ISL encapsulation**.

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

Click on the link below to get the full version.

Get How to Master CCNP TSHOOT Today



```
SwitchB#show interfaces fa0/15 switchport
Name: Fa0/15
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)Trunking Native Mode VLAN: 1 (default)
```

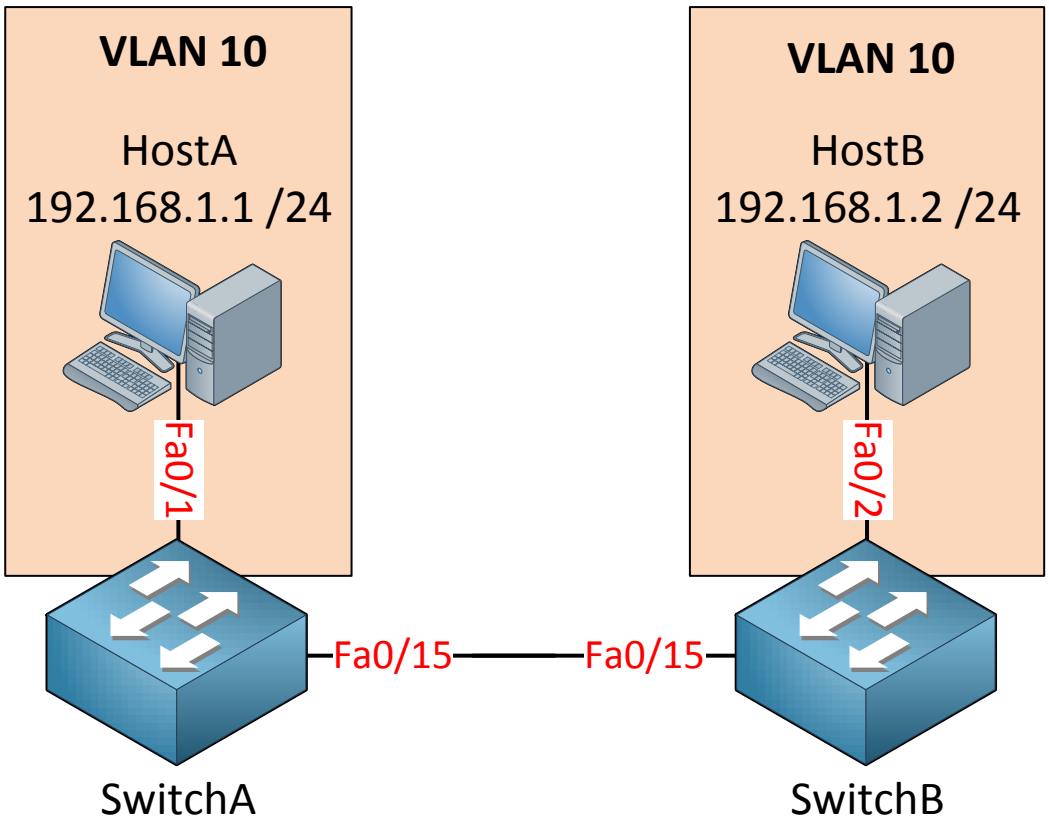
SwitchB is also in trunk mode but using **802.1Q encapsulation**. Be aware that (depending on the switch model) the default administrative mode might be **dynamic auto**. Two interfaces that are both running in dynamic auto mode will become an **access port**. It's best to change the interface to trunk mode by yourself. In our case both interfaces are trunking so that's good but we have an encapsulation protocol mismatch.

```
SwitchA(config)#interface fa0/15
SwitchA(config-if)#switchport trunk encapsulation dot1q
```

We'll change the encapsulation type so both switches are using 802.1Q.

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Problem solved! Our ping is working. **Lesson learned: Make sure you use the same encapsulation protocol when configuring trunks.**



Here's the same scenario again. I want to show you something else that is important to check when solving trunk issues. Assume we checked and verified that the following items are causing no issues:

- Interfaces (speed/duplex).
- Port-security.
- Switchport configuration (VLAN assignment, interface configured in access mode).

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Unfortunately the ping between the computers is still not working.

Let me show you the fa0/15 interfaces on the switches:

```
SwitchA#show interfaces fa0/15 switchport
Name: Fa0/15
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
```

```
SwitchB#show interfaces fa0/15 switchport
Name: Fa0/15
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
```

I'll verify that both interfaces are in trunk mode and that we are using the same encapsulation protocol (802.1Q). No problems here. Anything else that can go wrong with this trunk link? You bet!

```
SwitchA#show interfaces fa0/15 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/15      on               802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/15      20
```

```
SwitchB#show interfaces fa0/15 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/15      on               802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/15      20
```

The trunk might be operational but this doesn't mean that all VLANs are allowed over the trunk link. In the example above you can see that **only VLAN 20** is allowed.

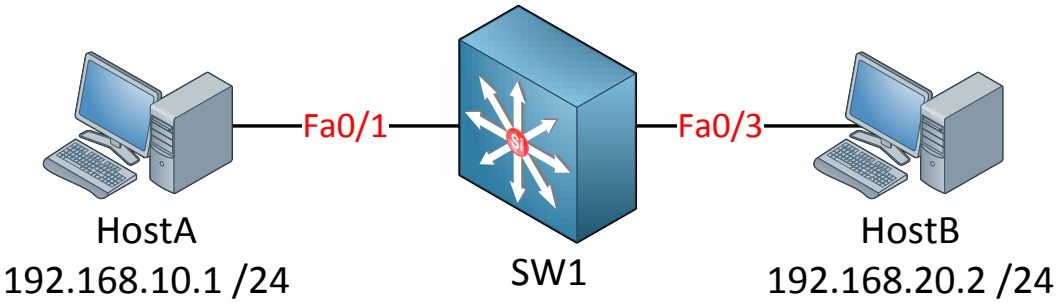
```
SwitchA(config)#interface fa0/15
SwitchA(config-if)#switchport trunk allowed vlan all
```

```
SwitchB(config)#interface fa0/15
SwitchB(config-if)#switchport trunk allowed vlan all
```

Let's allow all VLANs to pass the trunk.

```
C:\Documents and Settings\HostA>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

VLAN 10 is now able to pass the trunk link between the two switches. As a result I can ping between the computers….another problem bites the dust! **Lesson learned: Always check if a trunk allows all VLANs or not.**



Here's a new scenario for you. Two computers but you can see they have different IP addresses. The switch has another icon because it's a multilayer switch. Since the computers are in different subnets we have to worry about routing.

```
C:\Documents and Settings\HostA>ping 192.168.20.2

Pinging 192.168.1.2 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

A quick ping from HostA to HostB shows us that the two computers can't reach each other.

Where should we start troubleshooting?

```
C:\Documents and Settings\HostA>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 192.168.10.1
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.10.254
```

This isn't a book about windows but we do need to pay attention to our hosts. Since the computers need "to get out of their own subnet" we have to verify that the default gateway IP address is ok and reachable.

```
C:\Documents and Settings\VMWare>ping 192.168.10.254

Pinging 192.168.10.254 with 32 bytes of data:

Reply from 192.168.10.254: bytes=32 time=3ms TTL=255
Reply from 192.168.10.254: bytes=32 time=1ms TTL=255
Reply from 192.168.10.254: bytes=32 time=2ms TTL=255
Reply from 192.168.10.254: bytes=32 time=1ms TTL=255

Ping statistics for 192.168.10.254:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 3ms, Average = 1ms
```

HostA is able to reach the default gateway so we at least know that hostA is working fine.

```
C:\Documents and Settings\HostB>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 192.168.20.2
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.20.254
```

Here's the IP configuration of hostB.

Let's see if we can reach the default gateway!

```
C:\Documents and Settings\HostB>ping 192.168.20.254

Pinging 192.168.20.254 with 32 bytes of data:

Reply from 192.168.20.254: bytes=32 time=4ms TTL=255
Reply from 192.168.20.254: bytes=32 time=2ms TTL=255
Reply from 192.168.20.254: bytes=32 time=2ms TTL=255
Reply from 192.168.20.254: bytes=32 time=1ms TTL=255

Ping statistics for 192.168.20.254:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 4ms, Average = 2ms
```

That's also working. We know that the computers are not the issue because they know how to get out of their own subnet and the default gateway is reachable. Time to check out the switch (we are network guys after all…).

```
SwitchA#show interfaces fa0/1 switchport | include VLAN
Access Mode VLAN: 10 (VLAN0010)
SwitchA#show interfaces fa0/3 switchport | include VLAN
Access Mode VLAN: 20 (VLAN0020)
```

We can see that hostA is in VLAN 10 and hostB is in VLAN 20. I didn't check if the interfaces were up/up because I was already able to ping the default gateway IP addresses. This proves that fa0/1 and fa0/3 are working but I didn't know yet to what VLAN they belong.

```
SwitchA#show ip int brief | include Vlan
Vlan1                  unassigned      YES TFTP   up
down
Vlan10                 192.168.10.254  YES manual up                    up
Vlan20                 192.168.20.254  YES manual up                    up
```

Two SVI interfaces have been configured. These are the IP addresses that the computers use as default gateway. So why isn't our switch routing traffic?

```
SwitchA#show ip route
Default gateway is not set

Host               Gateway           Last Use    Total Uses  Interface
ICMP redirect cache is empty
```

Having IP addresses on interfaces doesn't automatically mean that we are going to route traffic. In order to do so we require a routing table. This switch doesn't have one…

```
SwitchA(config)#ip routing
```

Let's enable routing on this switch.

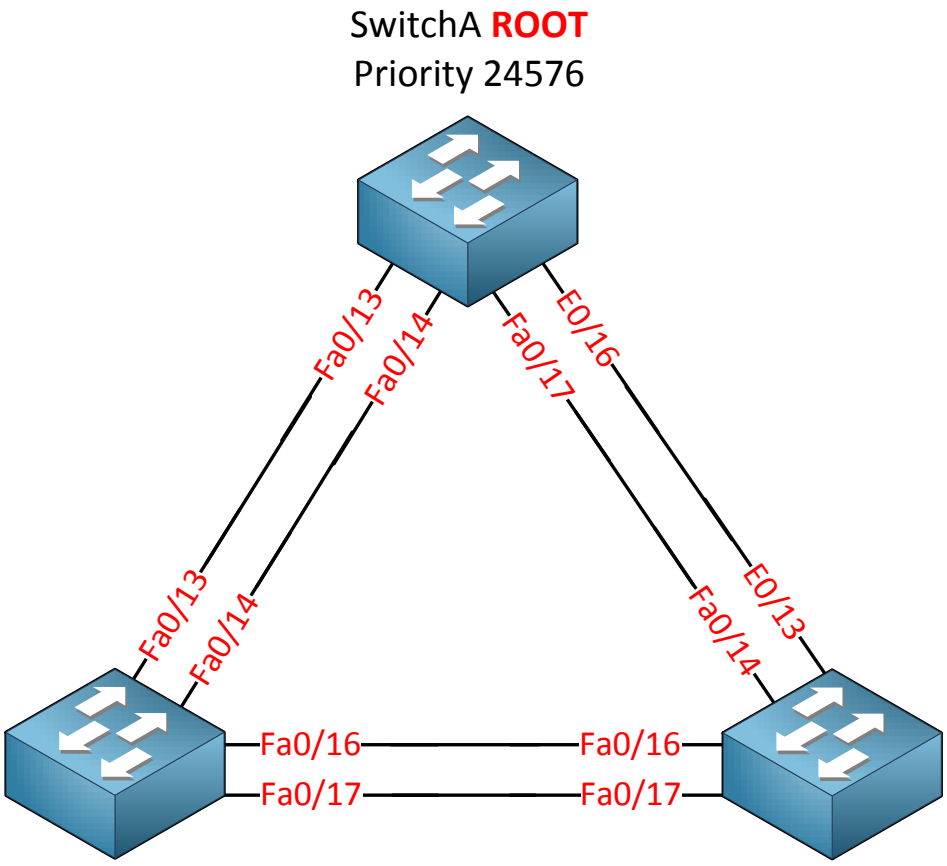
```
SwitchA#show ip route connected
C    192.168.10.0/24 is directly connected, Vlan10
C    192.168.20.0/24 is directly connected, Vlan20
```

This is looking better. The switch now knows where to forward IP packets to.

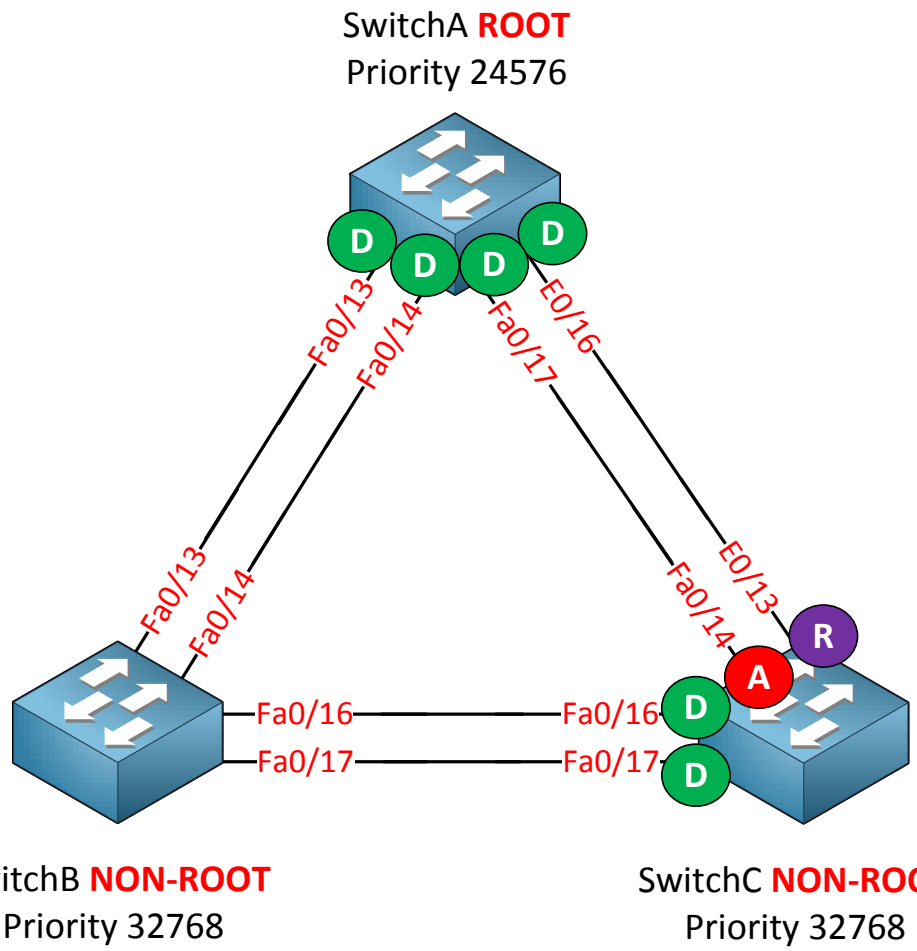
```
C:\Documents and Settings\HostA>ping 192.168.20.2

Pinging 192.168.20.2 with 32 bytes of data:

Reply from 192.168.20.2: bytes=32 time<1ms TTL=127
Reply from 192.168.20.2: bytes=32 time<1ms TTL=127
Reply from 192.168.20.2: bytes=32 time<1ms TTL=127
Reply from 192.168.20.2: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.20.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

There we go…the two computers can now reach each other! Problem solved ☺ **Lesson learned: If you use a multilayer switch for interVLAN routing make sure the SVI interfaces are configured correctly and that routing is enabled.**

You have now seen the most common errors that can happen with our interfaces, VLANs, trunks and routing issues when using multilayer switches. In the next part of this chapter we'll take a look at spanning-tree. Spanning-tree is a pretty robust protocol but there are a number of things that could go wrong, maybe you expect a certain output but your switches are telling you something different. Also because of misconfiguration some funky things can happen…let's take a look at some scenarios:



Here's a topology for you. Three switches and between the switches we have two links for redundancy. SwitchA has been elected as the root bridge for VLAN 1. When you are dealing with spanning-tree it's best to draw a small picture of the network and write down the interface roles for each switch (designated, non-designated/alternate or blocked). Note that one of the links between SwitchA and SwitchC is an Ethernet interface (10Mbit). All the other links are FastEthernet.



I used the show spanning-tree command to verify the interface roles for SwitchA and SwitchC. As you can see SwitchC has elected its Ethernet 0/13 interface as its root port and the FastEthernet 0/14 interface is elected as an alternate port. This is not a very good idea. It means we'll send all traffic down the 10Mbit link while the 100Mbit is not used at all. When a switch has to elect a root port it will select one like this:

1. Choose the interface that has the lowest cost to the root bridge.
2. If the cost is equal, select the lowest interface number.

Normally the cost of an Ethernet interface is higher than FastEthernet so it should select the FastEthernet interface.

Why did SwitchC pick the Ethernet 0/13 interface?

```
SwitchC#show spanning-tree vlan 1

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    24577
              Address     0011.bb0b.3600
              Cost        19
              Port        13 (FastEthernet0/13)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32769  (priority 32768 sys-id-ext 1)
              Address     000f.34ca.1000
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 15

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------
-----
E0/13               Root FWD 19        128.13   P2p
Fa0/14              Altn BLK 19        128.14   P2p
Fa0/16              Desg FWD 19        128.16   P2p
Fa0/17              Desg FWD 19        128.17   P2p
```

We can see that the Ethernet 0/13 and FastEthernet 0/14 interface have the same cost. SwitchC will then select the lowest interface number which is interface Ethernet 0/13.

```
SwitchC#show run interface fa0/13
Building configuration...

Current configuration : 102 bytes
!
Interface Ethernet0/13
 switchport mode dynamic desirable
 spanning-tree cost 19
```

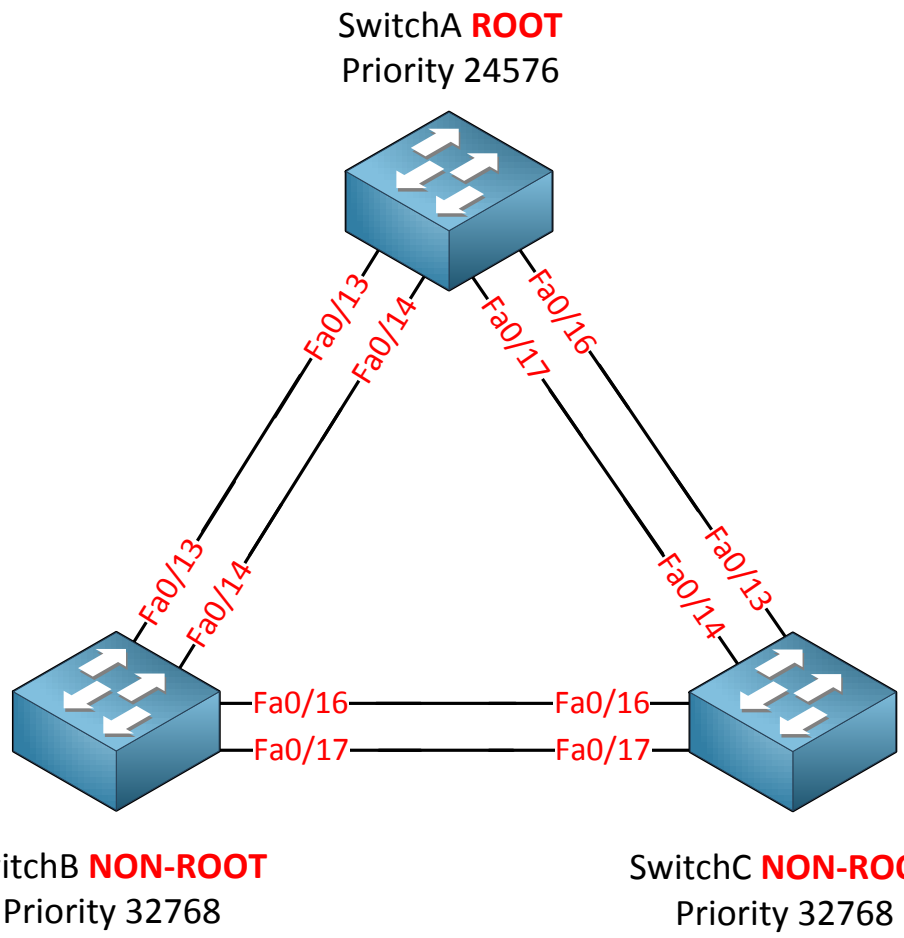
We'll check the interface configuration and you can see that someone has changed the cost of the interface to 19 (the default for FastEthernet interfaces).

```
SwitchC(config)#interface Ethernet 0/13
SwitchC(config-if)#no spanning-tree cost 19
```

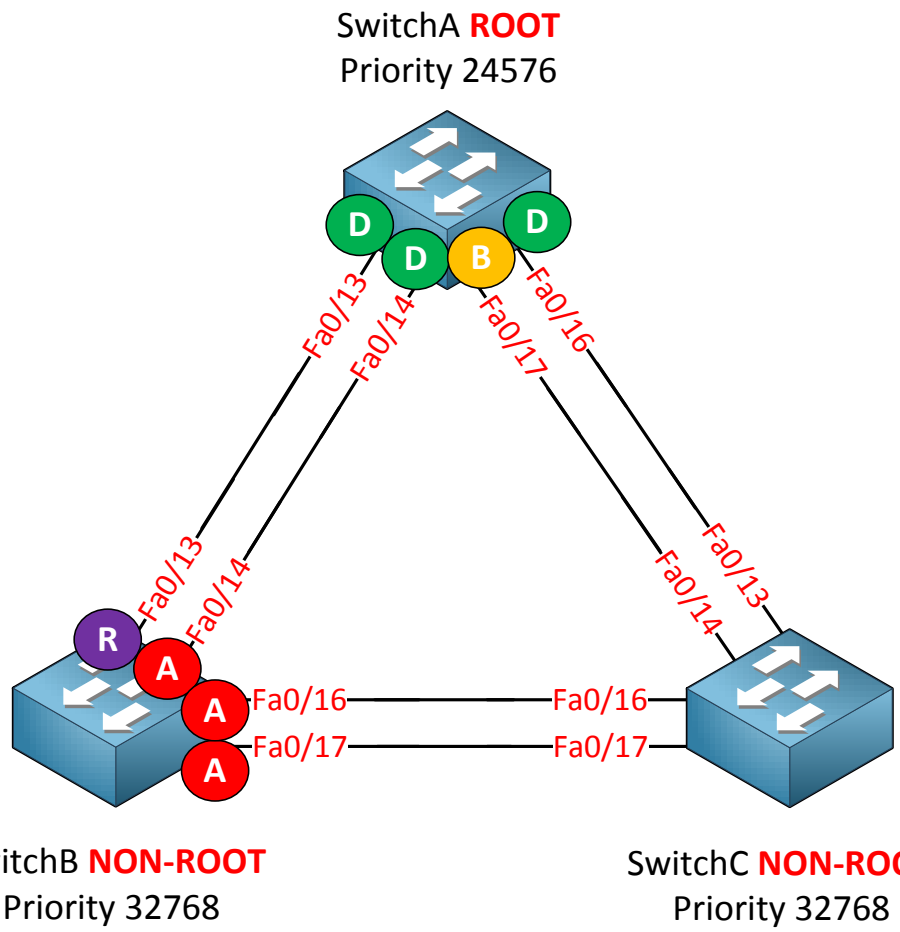
Let's get rid of the cost command.

```
SwitchC#show spanning-tree vlan 1

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    24577
              Address     0011.bb0b.3600
              Cost        19
              Port        14 (FastEthernet0/14)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32769  (priority 32768 sys-id-ext 1)
              Address     000f.34ca.1000
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 15

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------
-----
E0/13               Altn BLK 100       128.13   P2p
Fa0/14              Root FWD 19        128.14   P2p
Fa0/16              Desg FWD 19        128.16   P2p
Fa0/17              Desg FWD 19        128.17   P2p
```

After we removed the cost command you can see that the port state has changed. FastEthernet 0/14 is now the root port and the cost of the Ethernet 0/13 interface is 100 (which is the default for Ethernet interfaces). Problem solved! **Lesson learned: Make sure the interface you want to be the root port has the lowest cost path.**



Here's a new scenario for you. All the interfaces are equal (FastEthernet). SwitchA is the root bridge for VLAN 10 and after checking the interface roles this is what we find:



Hmm interesting…SwitchA is the root bridge and FastEthernet 0/17 has been elected as a **backup port**. Now that's something you see every day. SwitchB has elected a root port and all the other interfaces are alternate ports. I don't see anything on SwitchC.

```
SwitchA#show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol ieee
```

```
SwitchB#show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol ieee
```

```
SwitchC#show spanning-tree vlan 10

Spanning tree instance(s) for vlan 10 does not exist.
```

We can see that SwitchA and SwitchB are running spanning-tree for VLAN 10. SwitchC however is not running spanning-tree for VLAN 10. What could be the issue?

```
SwitchC#show interfaces fa0/13 | include line protocol
FastEthernet0/13 is up, line protocol is up (connected)
```

```
SwitchC#show interfaces fa0/14 | include line protocol
FastEthernet0/14 is up, line protocol is up (connected)
```

```
SwitchC#show interfaces fa0/16 | include line protocol
FastEthernet0/16 is up, line protocol is up (connected)
```

```
SwitchC#show interfaces fa0/17 | include line protocol
FastEthernet0/16 is up, line protocol is up (connected)
```

Of course it's not a bad idea to check if the interfaces on SwitchC are working or not (but of course this is something that you already learned and did in the first part of this chapter ☺).

```
SwitchC#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/13      desirable        n-isl          trunking      1
Fa0/14      desirable        n-isl          trunking      1
Fa0/16      desirable        n-isl          trunking      1
Fa0/17      desirable        n-isl          trunking      1

Port        Vlans allowed on trunk
Fa0/13      1-4094
Fa0/14      1-4094
Fa0/16      1-4094
Fa0/17      1-4094

Port        Vlans allowed and active in management domain
Fa0/13      1,10
Fa0/14      1,10
Fa0/16      1,10
Fa0/17      1,10
```

The interfaces are looking good. VLAN 10 is active on all interfaces of SwitchC. This means that spanning-tree should be active for VLAN 10.

```
SwitchC#show spanning-tree vlan 10

Spanning tree instance(s) for vlan 10 does not exist.
```

Let's take another look at this message. It says that spanning-tree for VLAN 10 does **not exist**. There are two reasons why could see this message:

- There are no interfaces active for VLAN 10.
- Spanning-tree has been disabled for VLAN 10.

We confirmed that VLAN 10 is active on all interfaces of SwitchC so maybe spanning-tree has been disabled globally?

```
SwitchC(config)#spanning-tree vlan 10
```

Let's give it a shot by typing in **spanning-tree vlan 10.**

```
SwitchC#show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol ieee
  Root ID     Priority    24586
              Address     0011.bb0b.3600
              Cost        19
              Port        13 (FastEthernet0/13)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32778  (priority 32768 sys-id-ext 10)
              Address     000f.34ca.1000
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------
-----
Fa0/13              Root FWD 19        128.13   P2p
Fa0/14              Altn BLK 19        128.14   P2p
Fa0/16              Desg FWD 19        128.16   P2p
Fa0/17              Desg FWD 19        128.17   P2p
```

There we go...that's looking better! Spanning-tree is now enabled for VLAN 10 and is working…problem solved! This issue might sound a bit lame but I do see it every now and then in the real world. A scenario I encountered before is a customer that was told by their wireless vendor to disable spanning-tree for the interfaces that connect to the wireless access point. This is what the customer typed in on the switch:

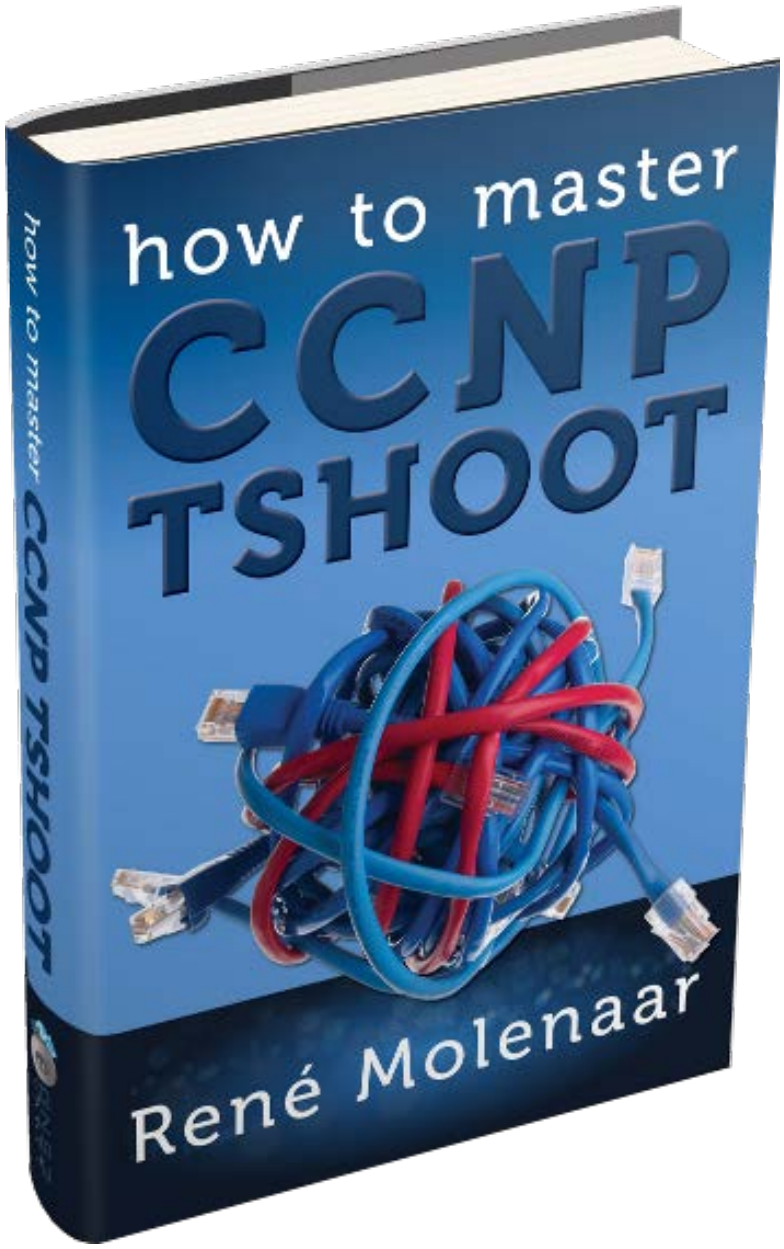
```
SwitchC(config)#interface fa0/1
SwitchC(config-if)#no spanning-tree vlan 10
SwitchC(config)#
```

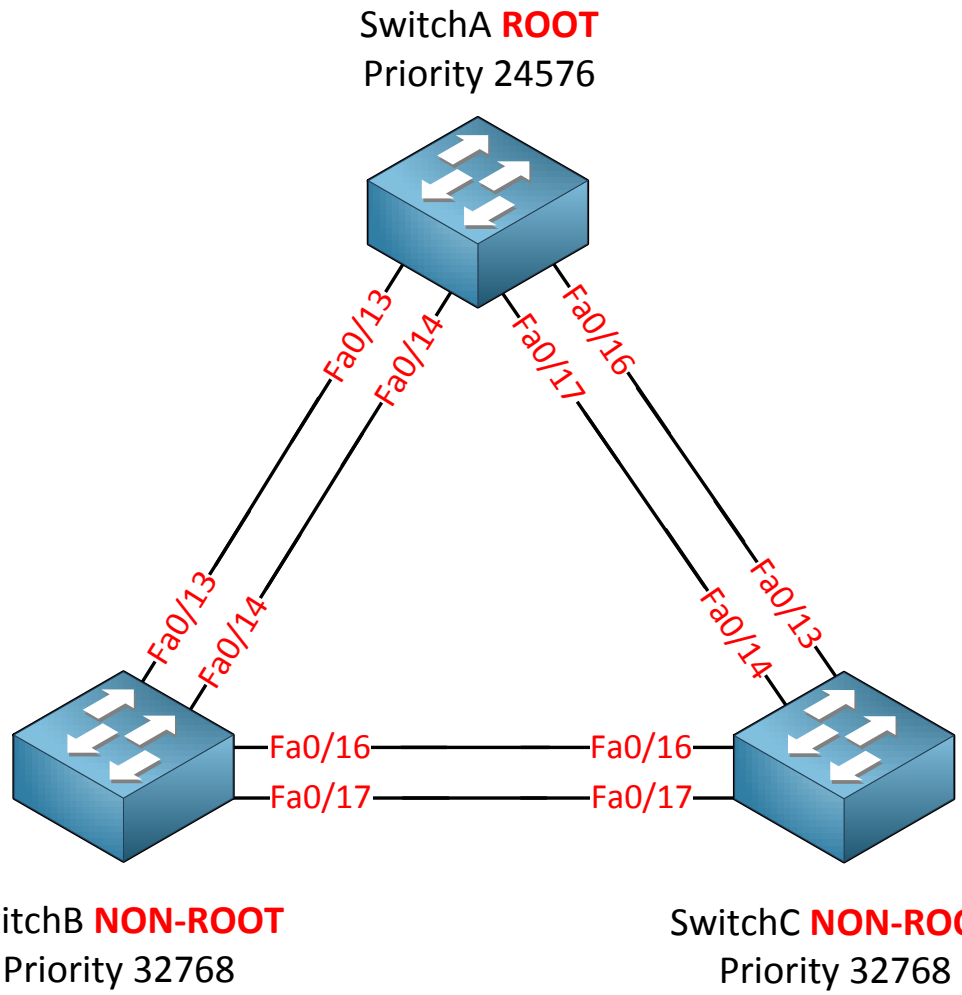
On the interface they typed **no spanning-tree vlan 10** but you can see you end up in the **global configuration mode**. There is no command to disable spanning-tree on the interface like this so the switch thinks you typed in the global command to disable spanning-tree. The switch accepts the command, disabled spanning-tree for VLAN 10 and kicks you back to global configuration mode…problem solved! Lesson learned: **Check if spanning-tree is enabled or disabled.**

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

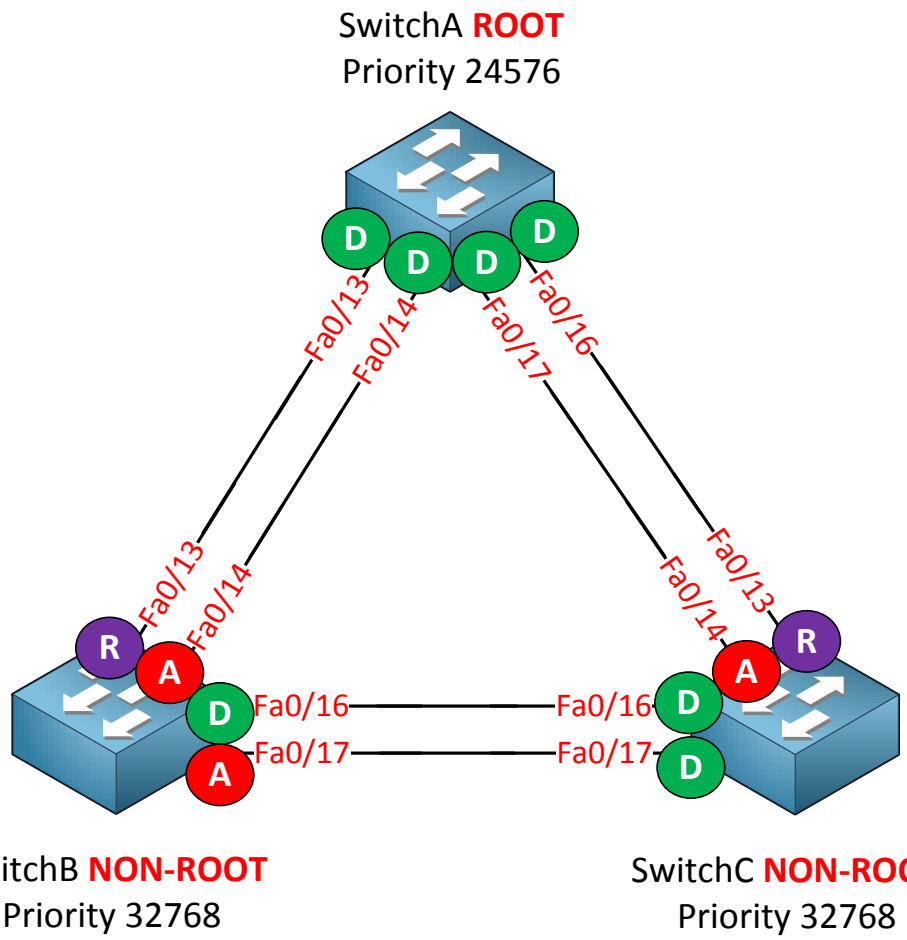
Click on the link below to get the full version.

Get How to Master CCNP TSHOOT Today





Let's continue with another scenario! Same topology…our customer however is complaining about bad performance. We'll start by verifying the roles of the interfaces:



Take a look at the picture above. Do you see that Interface FastEthernet 0/16 on SwitchB and SwitchC are designated? On SwitchA all interfaces are designated. What do you think happens once one of our switches forwards a broadcast or has to flood a frame? Bingo! We'll have a loop…

Normally in this topology the FastEthernet 0/16 and 0/17 interfaces on SwitchC should both be alternate ports because SwitchC has the worst bridge ID. Since they are both designated we can assume that SwitchC is not receiving BPDUs on these interfaces.

So why did spanning-tree fail here? An important detail to remember here is that spanning-tree **requires BPDUs sent between the switches** in order to **create a loop-free topology**. BPDUs can be filtered because of MAC access-lists, VLAN access-maps or maybe something from the spanning-tree toolkit?

```
SwitchA#show vlan access-map
```

```
SwitchB#show vlan access-map
```

```
SwitchC#show vlan access-map
```

There are no VLAN access maps on any of the switches.

```
SwitchA#show access-lists
```

```
SwitchB#show access-lists
```

```
SwitchC#show access-lists
```

There are no access-lists…

```
SwitchA#show port-security
Secure Port  MaxSecureAddr  CurrentAddr  SecurityViolation  Security Action
                (Count)       (Count)          (Count)
---------------------------------------------------------------------------
---------------------------------------------------------------------------
Total Addresses in System (excluding one mac per port)     : 0
Max Addresses limit in System (excluding one mac per port) : 6144
```

```
SwitchB#show port-security
Secure Port  MaxSecureAddr  CurrentAddr  SecurityViolation  Security Action
                (Count)       (Count)          (Count)
---------------------------------------------------------------------------
---------------------------------------------------------------------------
Total Addresses in System (excluding one mac per port)     : 0
Max Addresses limit in System (excluding one mac per port) : 6144
```

```
SwitchC#show port-security
Secure Port  MaxSecureAddr  CurrentAddr  SecurityViolation  Security Action
                (Count)       (Count)          (Count)
---------------------------------------------------------------------------
---------------------------------------------------------------------------
Total Addresses in System (excluding one mac per port)     : 0
Max Addresses limit in System (excluding one mac per port) : 6144
```

There's no port security…what about spanning-tree related commands?

```
SwitchB#show spanning-tree interface fa0/16 detail | include filter
   Bpdu filter is enabled
```

```
SwitchB#show spanning-tree interface fa0/17 detail | include filter
   Bpdu filter is enabled
```

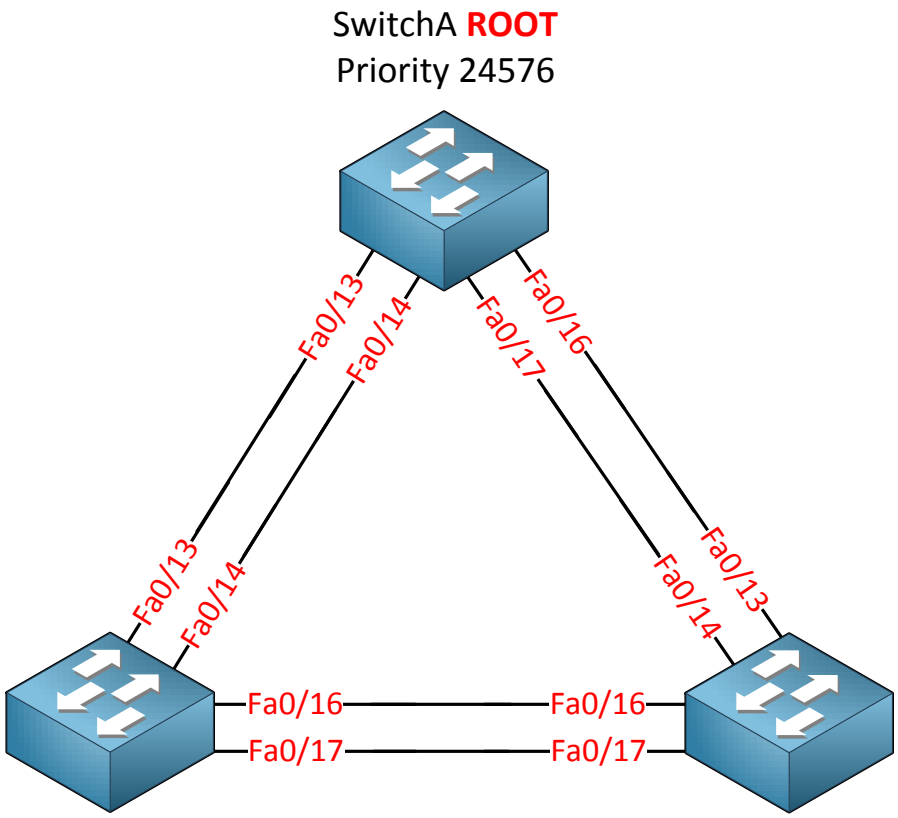
We found something! BPDU filter has been enabled on the FastEthernet 0/16 and 0/17 interfaces of SwitchB. Because of this SwitchC doesn't receive BPDUs from SwitchB.

```
SwitchB(config)#interface fa0/16
SwitchB(config-if)#no spanning-tree bpdufilter enable
SwitchB(config-if)#interface fa0/17
SwitchB(config-if)#no spanning-tree bpdufilter enable
```

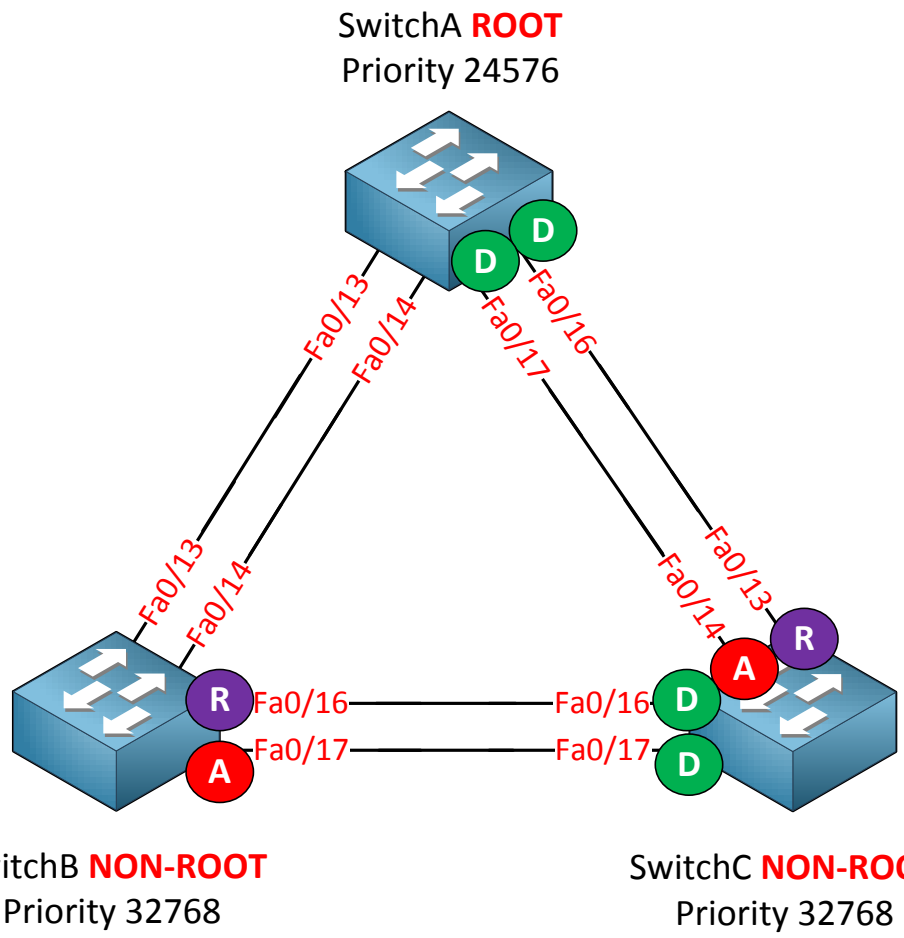
Let's get rid of the BPDU filter.

```
SwitchC#show spanning-tree vlan 10 | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------
-----
Fa0/13              Root FWD 19        128.13   P2p
Fa0/14              Altn BLK 19        128.14   P2p
Fa0/16              Altn BLK 19        128.16   P2p
Fa0/17              Altn BLK 19        128.17   P2p
```

Now you can see that FastEthernet 0/16 and 0/17 are both alternate ports and blocking traffic. Our topology is now loop-free…problem solved! **Lesson learned: make sure BPDUs are not blocked or filtered between switches.**



Here's a new topology for you. SwitchA has been elected as the root bridge for VLAN 10. All the interfaces are FastEthernet links.



After using the **show spanning-tree vlan 10** command this is what we see. All interfaces are equal but for some reason SwitchB decided to select FastEthernet 0/16 as its root port. Don't you agree that FastEthernet 0/13 should be the root port? It has a lower cost to reach the root bridge than FastEthernet 0/16.

```
SwitchB#show spanning-tree interface fa0/13

Vlan                 Role Sts Cost      Prio.Nbr Type
-------------------- ---- --- --------- -------- --------------------------
-----
VLAN0001             Root FWD 19        128.15   P2p
```

```
SwitchB#show spanning-tree interface fa0/14

Vlan                 Role Sts Cost      Prio.Nbr Type
-------------------- ---- --- --------- -------- --------------------------
-----
VLAN0001             Altn BLK 19        128.16   P2p
```

We can use the **show spanning-tree interface** command to check the spanning-tree information per interface. As you can see there's only a spanning-tree for VLAN 1 active on interface FastEthernet 0/13 and 0/14.

There are a number of things we could check to see what is going on:

```
SwitchB#show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol ieee
  Root ID     Priority    24586
              Address     0011.bb0b.3600
              Cost        38
              Port        18 (FastEthernet0/16)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32778  (priority 32768 sys-id-ext 10)
              Address     0019.569d.5700
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------
-----
Fa0/16              Root FWD 19        128.18   P2p
Fa0/17              Altn BLK 19        128.19   P2p
```

First it's always a good idea to check if spanning-tree is active for a certain VLAN. It's possible to disable spanning-tree by using the **no spanning-tree vlan X** command. In this scenario spanning-tree is active for VLAN 10 because we can see FastEthernet 0/16 and 0/17.

```
SwitchB#show interfaces fa0/13 switchport
Name: Fa0/13
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
```

```
SwitchB#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
```

We know that spanning-tree is active globally for VLAN 10 but that doesn't mean it's active on all interfaces. I can use the **show interfaces switchport** command to check if VLAN 10 is running on interface FastEthernet 0/13 and 0/14. This reveals us some interesting information. You can see that these interfaces ended up in **access mode** and they are in VLAN 1.

```
SwitchB(config)#interface fa0/13
SwitchB(config-if)#switchport mode trunk
```

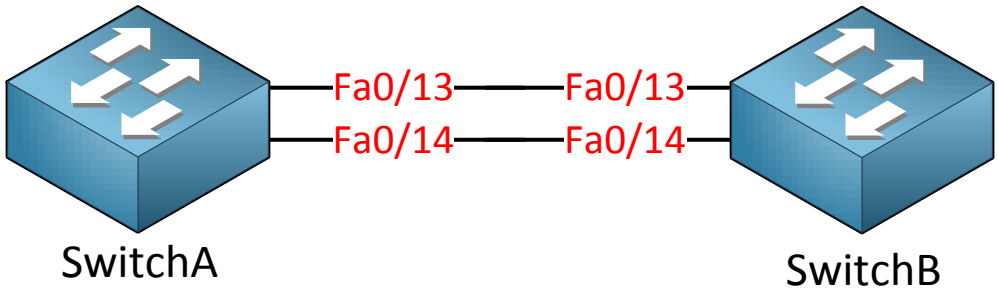
```
SwitchB(config-if)#interface fa0/14
SwitchB(config-if)#switchport mode trunk
```

Let's change the interfaces to trunks so VLAN 10 traffic can flow through these interfaces.

```
SwitchB#show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol ieee
  Root ID     Priority    24586
              Address     0011.bb0b.3600
              Cost        19
              Port        15 (FastEthernet0/13)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32778  (priority 32768 sys-id-ext 10)
              Address     0019.569d.5700
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------
-----
Fa0/13              Root FWD 19        128.15   P2p
Fa0/14              Altn BLK 19        128.16   P2p
Fa0/16              Altn BLK 19        128.18   P2p
Fa0/17              Altn BLK 19        128.19   P2p
```

This is looking better. VLAN 10 traffic now runs on interface FastEthernet 0/13 and 0/14 and you can see that interface FastEthernet 0/13 is now elected as the root port. Problem solved! **Lesson learned: make sure the VLAN is active on the interface before looking at spanning-tree related issues.**

These are all the spanning-tree issues I wanted to show you. Before you start looking at spanning-tree commands you should verify that all your interfaces are operational and that the VLANs are passing your trunk links. Spanning-tree relies on BPDUs that are exchanged between the switches so make sure these are not filtered. If you are still facing issues you should check the spanning-tree configuration itself.
The last part of this chapter will be about troubleshooting Etherchannels. Most of the issues with etherchannels are because of **misconfiguration**.



In this scenario there are only two switches and two interfaces. The idea is to form an

etherchannel by bundling the FastEthernet 0/13 and 0/14 interfaces but this is not working…

```
SwitchA#show interfaces fa0/13 | include line protocol
FastEthernet0/13 is up, line protocol is up (connected)
```

```
SwitchA#show interfaces fa0/14 | include line protocol
FastEthernet0/14 is up, line protocol is up (connected)
```

```
SwitchB#show interfaces fa0/13 | include line protocol
FastEthernet0/13 is up, line protocol is up (connected)
```

```
SwitchB#show interfaces fa0/14 | include line protocol
FastEthernet0/14 is up, line protocol is up (connected)
```

First I'll check if the interfaces are all up and running, this seems to be the case.

```
SwitchA#show ip int brief | include Port
Port-channel1          unassigned      YES unset  down
down
```

```
SwitchB#show ip int brief | include Port
Port-channel1          unassigned      YES unset  down
down
```

We can verify that a port-channel interface has been created but it is down.

```
SwitchA#show etherchannel summary | begin Group
Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
-----
1      Po1(SD)         LACP      Fa0/13(I)   Fa0/14(I)
```

```
SwitchB#show etherchannel summary | begin Group
Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
-----
1      Po1(SD)         PAgP      Fa0/13(I)   Fa0/14(I)
```

Here's a nice command to verify your etherchannel. Use the **show etherchannel summary** to see your port-channels. We can see that SwitchA has been configured for LACP and SwitchB for PAgP, this is never going to work.

```
SwitchA#show etherchannel 1 detail
Group state = L2
Ports: 2   Maxports = 16
Port-channels: 1 Max Port-channels = 16
Protocol:   LACP
Minimum Links: 0
       Ports in the group:
       -------------------
Port: Fa0/13
------------

Port state    = Up Sngl-port-Bndl Mstr Not-in-Bndl
Channel group = 1           Mode = Active          Gcchange = -
Port-channel  = null        GC   =   -             Pseudo port-channel = Po1
Port index    = 0           Load = 0x00            Protocol =   LACP

Flags:  S - Device is sending Slow LACPDUs   F - Device is sending fast LACPDUs.
        A - Device is in active mode.        P - Device is in passive mode.

Local information:
                            LACP port     Admin     Oper    Port         Port
Port      Flags   State     Priority      Key       Key     Number       State
Fa0/13    SA      indep     32768         0x1       0x1     0xF          0x7D

Age of the port in the current state: 0d:00h:11m:59s

Port: Fa0/14
------------

Port state    = Up Sngl-port-Bndl Mstr Not-in-Bndl
Channel group = 1           Mode = Active          Gcchange = -
Port-channel  = null        GC   =   -             Pseudo port-channel = Po1
Port index    = 0           Load = 0x00            Protocol =   LACP

Flags:  S - Device is sending Slow LACPDUs   F - Device is sending fast LACPDUs.
        A - Device is in active mode.        P - Device is in passive mode.

Local information:
                            LACP port     Admin     Oper    Port         Port
Port      Flags   State     Priority      Key       Key     Number       State
Fa0/14    SA      indep     32768         0x1       0x1     0x10         0x7D

Age of the port in the current state: 0d:00h:12m:38s

               Port-channels in the group:
               ---------------------------

Port-channel: Po1    (Primary Aggregator)

------------

Age of the Port-channel   = 0d:00h:12m:55s
Logical slot/port   = 2/1          Number of ports = 0
HotStandBy port = null
Port state          = Port-channel Ag-Not-Inuse
Protocol            =   LACP
Port security       = Disabled
```

The best command to use is **show etherchannel detail**. This gives you a lot of information but I'm particularly interested in seeing if LACP is configured for **passive** or **active** mode. Interfaces in active mode will "actively" try to form an etherchannel. Interfaces in passive

mode will only respond to LACP requests.

```
SwitchB#show etherchannel 1 detail
Group state = L2
Ports: 2   Maxports = 8
Port-channels: 1 Max Port-channels = 1
Protocol:   PAgP
Minimum Links: 0
      Ports in the group:
      -------------------
Port: Fa0/13
------------

Port state    = Up Sngl-port-Bndl Mstr Not-in-Bndl
Channel group = 1           Mode = Desirable-Sl    Gcchange = 0
Port-channel  = null        GC   = 0x00010001      Pseudo port-channel = Po1
Port index    = 0           Load = 0x00            Protocol =   PAgP

Flags:  S - Device is sending Slow hello.  C - Device is in Consistent state.
        A - Device is in Auto mode.        P - Device learns on physical port.
        d - PAgP is down.
Timers: H - Hello timer is running.        Q - Quit timer is running.
        S - Switching timer is running.    I - Interface timer is running.

Local information:
                              Hello    Partner  PAgP     Learning  Group
Port      Flags State   Timers  Interval Count   Priority   Method  Ifindex
Fa0/13     U4/S4   H   30s  0        128        Any      10013

Age of the port in the current state: 0d:00h:15m:25s

Port: Fa0/14
------------

Port state    = Up Sngl-port-Bndl Mstr Not-in-Bndl
Channel group = 1           Mode = Desirable-Sl    Gcchange = 0
Port-channel  = null        GC   = 0x00010001      Pseudo port-channel = Po1
Port index    = 0           Load = 0x00            Protocol =   PAgP

Flags:  S - Device is sending Slow hello.  C - Device is in Consistent state.
        A - Device is in Auto mode.        P - Device learns on physical port.
        d - PAgP is down.
Timers: H - Hello timer is running.        Q - Quit timer is running.
        S - Switching timer is running.    I - Interface timer is running.

Local information:
                              Hello    Partner  PAgP     Learning  Group
Port      Flags State   Timers  Interval Count   Priority   Method  Ifindex
Fa0/14     U4/S4   H   30s  0        128        Any      10014

Age of the port in the current state: 0d:00h:15m:28s

               Port-channels in the group:
               ---------------------------

Port-channel: Po1
------------

Age of the Port-channel   = 0d:00h:15m:51s
Logical slot/port   = 2/1          Number of ports = 0
GC                  = 0x00000000      HotStandBy port = null
Port state          = Port-channel Ag-Not-Inuse
```

```
Protocol            =   PAgP
Port security       = Disabled
```

Here's the output of show etherchannel detail of SwitchB. We can see that it has been configured for PAgP and the interfaces are configured for **desirable** mode. If they would have been configured for **auto** mode we would see the **A** flag.

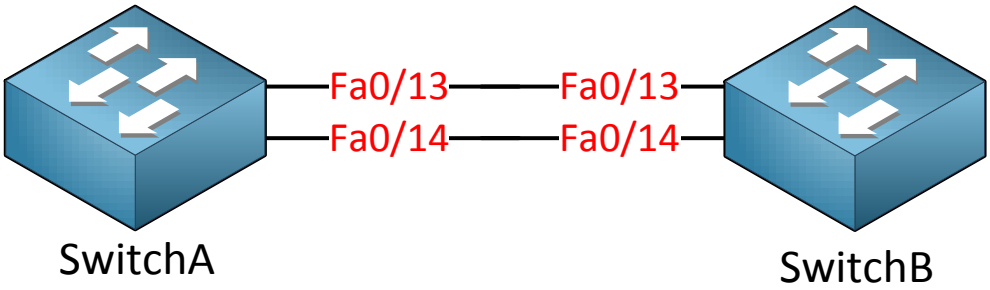
```
SwitchB(config)#no interface po1
SwitchB(config)#interface fa0/13
SwitchB(config-if)#channel-group 1 mode passive
SwitchB(config-if)#exit
SwitchB(config)#interface fa0/14
SwitchB(config-if)#channel-group 1 mode passive
```

Let's get rid of the port-channel interface first, if we don't do this you'll see an error when you try to change the channel-group mode on the interfaces.

```
SwitchA# %LINK-3-UPDOWN: Interface Port-channel1, changed state to up
```

```
SwitchB# %LINK-3-UPDOWN: Interface Port-channel1, changed state to up
```

After changing the configuration we see the port-channel1 going up. Problem solved!
Lesson learned: Make sure you use the same EtherChannel mode on both sides.



Time for another error! Same topology and an etherchannel that is not functioning:

```
SwitchA#show ip interface brief | include Port
Port-channel1          unassigned      YES unset  down
down
```

```
SwitchB#show ip interface brief | include Port
Port-channel1          unassigned      YES unset  down
down
```

We can verify that the port-channel interface exists but it's down on both sides.

```
SwitchA#show etherchannel 1 summary | begin Group
Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
-----
1      Po1(SD)         PAgP      Fa0/13(D)   Fa0/14(D)
```

```
SwitchB#show etherchannel 1 summary | begin Group
Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
-----
1      Po1(SD)         PAgP      Fa0/13(D)   Fa0/14(D)
```

We can also see that interface FastEthernet 0/13 and 0/14 have both been added to the port-channel interface.

```
SwitchA#show ip interface brief | begin FastEthernet0/13
FastEthernet0/13       unassigned      YES unset  up                    up
FastEthernet0/14       unassigned      YES unset  up                    up
```

```
SwitchB#show ip interface brief | begin FastEthernet0/13
FastEthernet0/13       unassigned      YES unset  up                    up
FastEthernet0/14       unassigned      YES unset  up                    up
```

The FastEthernet interfaces are looking good so we know this is not the issue. Let's dive deeper into the etherchannel configuration.

```
SwitchA#show etherchannel 1 port
      Ports in the group:
      -------------------
Port: Fa0/13
------------

Port state    = Up Sngl-port-Bndl Mstr Not-in-Bndl
Channel group = 1           Mode = Automatic-Sl    Gcchange = 0
Port-channel  = null        GC   = 0x00010001      Pseudo port-channel =
Po1
Port index    = 0           Load = 0x00            Protocol =   PAgP

Flags:  S - Device is sending Slow hello.  C - Device is in Consistent
state.
        A - Device is in Auto mode.        P - Device learns on physical
port.
        d - PAgP is down.
Timers: H - Hello timer is running.        Q - Quit timer is running.
        S - Switching timer is running.    I - Interface timer is running.

Local information:
                              Hello    Partner  PAgP     Learning  Group
Port      Flags State   Timers  Interval Count   Priority   Method  Ifindex
Fa0/13    A  U2/S4      1s   0        128        Any      10013

Age of the port in the current state: 0d:00h:11m:38s
```

```
Port: Fa0/14
------------

Port state    = Up Sngl-port-Bndl Mstr Not-in-Bndl
Channel group = 1           Mode = Automatic-Sl    Gcchange = 0
Port-channel  = null        GC   = 0x00010001      Pseudo port-channel =
Po1
Port index    = 0           Load = 0x00            Protocol =   PAgP

Flags:  S - Device is sending Slow hello.  C - Device is in Consistent
state.
        A - Device is in Auto mode.        P - Device learns on physical
port.
        d - PAgP is down.
Timers: H - Hello timer is running.        Q - Quit timer is running.
        S - Switching timer is running.    I - Interface timer is running.

Local information:
                                Hello    Partner  PAgP     Learning  Group
Port      Flags State   Timers  Interval Count   Priority   Method  Ifindex
Fa0/14    A  U2/S4      1s   0        128        Any      10014

Age of the port in the current state: 0d:00h:11m:41s
```

We can see that FastEthernet 0/13 and 0/14 on SwitchA are both configured for PAgP Auto mode (because of the "A" flag).

```
SwitchB#show etherchannel 1 port
      Ports in the group:
      -------------------
Port: Fa0/13
------------

Port state    = Up Sngl-port-Bndl Mstr Not-in-Bndl
Channel group = 1           Mode = Automatic-Sl    Gcchange = 0
Port-channel  = null        GC   = 0x00010001      Pseudo port-channel =
Po1
Port index    = 0           Load = 0x00            Protocol =   PAgP

Flags:  S - Device is sending Slow hello.  C - Device is in Consistent
state.
        A - Device is in Auto mode.        P - Device learns on physical
port.
        d - PAgP is down.
Timers: H - Hello timer is running.        Q - Quit timer is running.
        S - Switching timer is running.    I - Interface timer is running.

Local information:
                                Hello    Partner  PAgP     Learning  Group
Port      Flags State   Timers  Interval Count   Priority   Method  Ifindex
Fa0/13    A  U2/S4      1s   0        128        Any      10013

Age of the port in the current state: 0d:00h:13m:13s

Port: Fa0/14
------------
```

```
Port state    = Up Sngl-port-Bndl Mstr Not-in-Bndl
Channel group = 1           Mode = Automatic-Sl    Gcchange = 0
Port-channel  = null        GC   = 0x00010001      Pseudo port-channel =
Po1
Port index    = 0           Load = 0x00            Protocol =   PAgP

Flags:  S - Device is sending Slow hello.  C - Device is in Consistent
state.
        A - Device is in Auto mode.        P - Device learns on physical
port.
        d - PAgP is down.
Timers: H - Hello timer is running.        Q - Quit timer is running.
        S - Switching timer is running.    I - Interface timer is running.

Local information:
                                Hello    Partner  PAgP     Learning  Group
Port      Flags State   Timers  Interval Count    Priority   Method  Ifindex
Fa0/14    A   U2/S4     1s   0        128        Any      10014

Age of the port in the current state: 0d:00h:13m:13s
```

FastEthernet 0/13 and 0/14 on SwitchB are also configured for PAgP auto. This is never going to work because both switches are now waiting passively for PAgP messages.

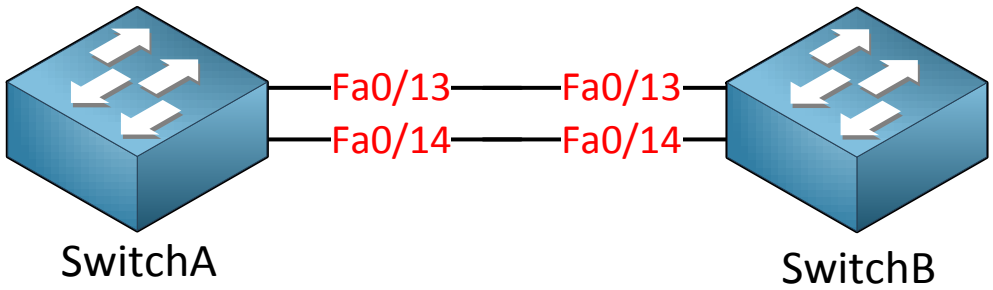
```
SwitchB(config)#interface fa0/13
SwitchB(config-if)#channel-group 1 mode desirable
SwitchB(config-if)#interface fa0/14
SwitchB(config-if)#channel-group 1 mode desirable
```

Let's change one of the switches so it will actively send PAgP messages.

```
SwitchA# %LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel1,
changed state to up
```

```
SwitchB# %LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel1,
changed state to up
```

The etherchannel is now working…problem solved! **Lesson learned: When using PAgP make sure at least one of the switches is using desirable mode or in case of LACP make sure one switch is in active mode.**



One last scenario for you: An etherchannel is configured between SwitchA and SwitchB but

the customer is complaining that the link is slow…what could possibly be wrong?

```
SwitchA#show ip int brief | include Port
Port-channel1          unassigned      YES unset  up                    up
```

```
SwitchB#show ip int brief | include Port
Port-channel1          unassigned      YES unset  up                    up
```

A quick check tells us that the port-channel interface is operational.

```
SwitchA#show etherchannel 1 detail
Group state = L2
Ports: 2   Maxports = 8
Port-channels: 1 Max Port-channels = 1
Protocol:   PAgP
Minimum Links: 0
       Ports in the group:
       -------------------
Port: Fa0/13
------------

Port state    = Up Cnt-bndl Suspend Not-in-Bndl
Channel group = 1           Mode = Automatic-Sl    Gcchange = 0
Port-channel  = null        GC   = 0x00000000      Pseudo port-channel = Po1
Port index    = 0           Load = 0x00            Protocol =   PAgP

Flags:  S - Device is sending Slow hello.  C - Device is in Consistent state.
        A - Device is in Auto mode.        P - Device learns on physical port.
        d - PAgP is down.
Timers: H - Hello timer is running.        Q - Quit timer is running.
        S - Switching timer is running.    I - Interface timer is running.

Local information:
                               Hello    Partner  PAgP     Learning  Group
Port      Flags State   Timers  Interval Count   Priority   Method  Ifindex
Fa0/13    dA   U1/S1       1s   0        128        Any      0

Age of the port in the current state: 0d:01h:10m:37s

Probable reason: speed of Fa0/13 is 100M, Fa0/14 is 10M
Port: Fa0/14
------------

Port state    = Up Mstr In-Bndl
Channel group = 1           Mode = Automatic-Sl    Gcchange = 0
Port-channel  = Po1         GC   = 0x00010001      Pseudo port-channel = Po1
Port index    = 0           Load = 0x00            Protocol =   PAgP

Flags:  S - Device is sending Slow hello.  C - Device is in Consistent state.
        A - Device is in Auto mode.        P - Device learns on physical port.
        d - PAgP is down.
Timers: H - Hello timer is running.        Q - Quit timer is running.
        S - Switching timer is running.    I - Interface timer is running.

Local information:
                               Hello    Partner  PAgP     Learning  Group
Port      Flags State   Timers  Interval Count   Priority   Method  Ifindex
Fa0/14    SAC  U6/S7    HQ  30s  1        128        Any      5001

Partner's information:
```

```
          Partner              Partner          Partner         Partner Group
Port      Name                 Device ID        Port       Age  Flags   Cap.
Fa0/14    SwitchB              0019.569d.5700  Fa0/14      15s SC  10001

Age of the port in the current state: 0d:00h:04m:29s

       Port-channels in the group:
       ---------------------------

Port-channel: Po1
------------

Age of the Port-channel   = 0d:01h:30m:23s
Logical slot/port   = 2/1          Number of ports = 1
GC                  = 0x00010001      HotStandBy port = null
Port state          = Port-channel Ag-Inuse
Protocol            =   PAgP
Port security       = Disabled

Ports in the Port-channel:

Index   Load   Port     EC state        No of bits
------+------+------+------------------+-----------
  0     00     Fa0/14   Automatic-Sl       0

Time since last port bundled:    0d:00h:04m:31s    Fa0/14
Time since last port Un-bundled: 0d:00h:08m:12s    Fa0/14
```

The show etherchannel detail command produces a lot of output but it does tell us what's going on. You can see that interface FastEthernet 0/13 and 0/14 both have been configured for this port-channel but the switch was unable to bundle them because FastEthernet 0/14 is configured for 10Mbit. You can see this in the **probable reason** that I highlighted.

```
SwitchB#show etherchannel 1 detail | include reason
Probable reason: speed of Fa0/13 is 100M, Fa0/14 is 10M
```

This is a good reason to use one of the operators for show command. I'm only interested in seeing the probably reason that the "show etherchannel detail" command produces.

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#speed auto
```

```
SwitchB(config)#interface fa0/14
SwitchB(config-if)#speed auto
```

Let's change the speed to auto. We need to make sure that FastEthernet 0/13 and 0/14 both have the same configuration. It's not a bad idea to do a show run to check if they have the same commands applied to them.

```
SwitchA# %LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel1,
changed state to down
%LINK-3-UPDOWN: Interface Port-channel1, changed state to down
%LINK-3-UPDOWN: Interface Port-channel1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel1, changed
state to up
```

```
SwitchB# %LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel1,
changed state to down
```

```
%LINK-3-UPDOWN: Interface Port-channel1, changed state to down
%LINK-3-UPDOWN: Interface Port-channel1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel1, changed
state to up
```

You will probably see a couple of messages about your interfaces bouncing down and up.

```
SwitchA#show etherchannel 1 summary | begin Group
Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------
-----
1      Po1(SU)         PAgP      Fa0/13(P)   Fa0/14(P)
```

```
SwitchB#show etherchannel 1 summary | begin Group
Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------
-----
1      Po1(SU)         PAgP      Fa0/13(P)   Fa0/14(P)
```

Now we see that both interfaces have been added to the port-channel...problem solved!
Lesson learned: Make sure all interfaces that will be added to the port-channel have the exact same configuration!

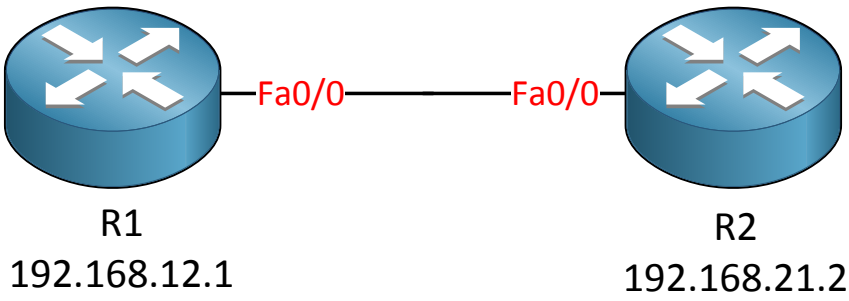
This is it…the end of the troubleshooting switching chapter. I showed you the most common errors for interfaces, vlans, trunks, spanning-tree and etherchannel. In the next chapter we'll move our way further up the OSI model and look at routing issues (EIGRP).

4. Troubleshooting EIGRP

In this chapter we'll take a look at Cisco's EIGRP routing protocol. EIGRP is an advanced distance vector routing protocol that has to establish a neighbor relationship before updates are sent. Because of this the first thing we'll have to do is check if the neighbor adjacency is working properly. If this is the case we can continue by checking if networks are being advertised or not. In this chapter I'll show you everything that can go wrong with EIGRP, how to fix it and in what order. Let's get started with the neighbor adjacency!

There are a number of items that cause problems with EIGRP neighbor adjacencies:

- **Uncommon subnet**: EIGRP neighbors with IP addresses that are not in the same subnet.
- **K value mismatch**: By default bandwidth and delay are enabled for the metric calculation. We can enable load and reliability as well but we have to do it on all EIGRP routers.
- **AS mismatch**: The autonomous system number has to match on both EIGRP routers in order to form a neighbor adjacency.
- **Layer 2 issues**: EIGRP works on layer 3 of the OSI-model. If layer 1 and 2 are not working properly we'll have issues with forming a neighbor adjacency.
- **Access-list issues**: It's possible that someone created an access-list that filters out multicast traffic. EIGRP by default uses 224.0.0.10 to communicate with other EIGRP neighbors.
- **NBMA**: Non Broadcast Multi Access networks like frame-relay will not allow broadcast or multicast traffic by default. This can prevent EIGRP from forming EIGRP neighbor adjacencies.




```
R1(config)#interface f0/0
R1(config-if)#ip address 192.168.12.1 255.255.255.0
R1(config-if)#router eigrp 12
R1(config-router)#network 192.168.12.0
```

```
R2(config)#interface f0/0
R2(config-if)#ip address 192.168.21.2 255.255.255.0
R2(config)#router eigrp 12
R2(config-router)#network 192.168.21.0
```

The **uncommon subnet** error is easy to spot. In the example above we have 2 routers and you can see I configured a different subnet on each interface.

After enabling EIGRP the following errors pops up:

```
R1# IP-EIGRP(Default-IP-Routing-Table:12): Neighbor 192.168.21.2 not on
common subnet for FastEthernet0/0
```

```
R2# IP-EIGRP(Default-IP-Routing-Table:12): Neighbor 192.168.12.1 not on
common subnet for FastEthernet0/0
```

Both routers complain that they are not on the same subnet.

```
R2(config-router)#interface f0/0
R2(config-if)#ip address 192.168.12.2 255.25
R2(config)#router eigrp 12
R2(config-router)#no network 192.168.21.0
R2(config-router)#network 192.168.12.0
```

I'll change the IP address on R2 and make sure the correct network command is configured for EIGRP.

```
R1# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2
(FastEthernet0/0) is up: new adjacency
```

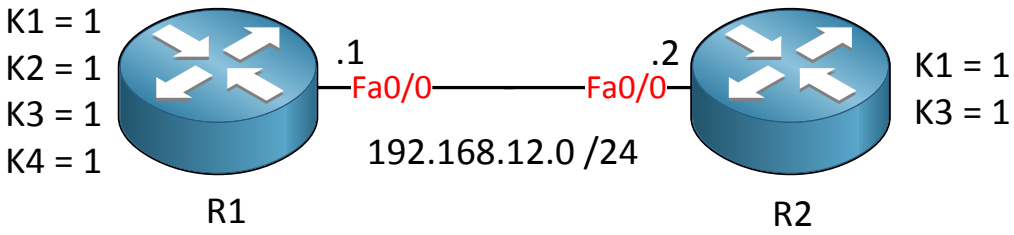
```
R2#  %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.1
(FastEthernet0/0) is up: new adjacency
```

Voila! We now have an EIGRP neighbor adjacency.

```
R1#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)       Cnt
Num
0   192.168.12.2            Fa0/0             13 00:05:15    3   200  0  3
```

```
R2#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)       Cnt
Num
0   192.168.12.1            Fa0/0             11 00:05:32 1263  5000  0  3
```

We can verify this by using the show ip eigrp neighbors command. **Lesson learned: Make sure both routers are on the same subnet.**



This time the IP addresses are correct but we are using different K values on both sides. R1 has enabled bandwidth, delay, load and reliability. R2 is only using bandwidth and delay.

```
R1# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2
(FastEthernet0/0) is down: K-value mismatch
```

```
R1# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2
(FastEthernet0/0) is down: Interface Goodbye received
```

This error is easy to spot because your console will give you the "K-value mismatch" error on both routers.

```
R1#show run | section eigrp
router eigrp 12
 network 192.168.12.0
 metric weights 0 1 1 1 1 0
 auto-summary
```

We can verify our configuration by looking at both routers. You can see the K values were changed on R1.

```
R2(config)#router eigrp 12
R2(config-router)#metric weights 0 1 1 1 1 0
```

We'll make sure the K values are the same on both routers, I'll change R2.

```
R1# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2
(FastEthernet0/0) is up: new adjacency
```

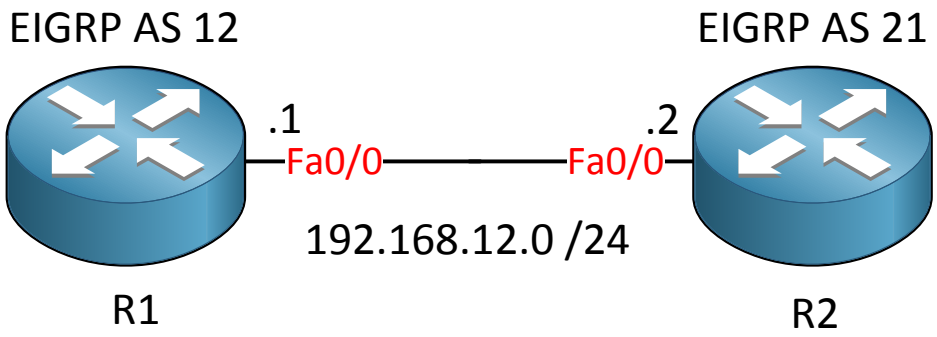
```
R2# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.1
(FastEthernet0/0) is up: new adjacency
```

After changing the K values we have an EIGRP neighbor adjacency.

```
R1#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)       Cnt
Num
0   192.168.12.2            Fa0/0             13 00:02:11   13   200  0  6
```

```
R2#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)       Cnt
Num
0   192.168.12.1            Fa0/0             13 00:02:42   19   200  0  6
```

Another problem solved! **Lesson learned: Make sure the K-values are the same on all EIGRP routers within the same autonomous system.** Let's continue with the next error…



Here's another example of a typical Monday morning problem. There's a mismatch in the AS number. When we configure EIGRP we have to type in an AS number. Unlike OSPF (which uses a process ID) this number has to be the same on both routers.

```
R1#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
```

```
R2#show ip eigrp neighbors
IP-EIGRP neighbors for process 21
```

Unlike the other EIGRP configuration mistakes this one doesn't produce an error message. We can use show ip eigrp neighbors and see that there are no neighbors. Use your eagle eyes to spot for differences and you'll quickly see that we are not using the same AS number.

```
R1#show run | section eigrp
router eigrp 12
 network 192.168.12.0
 auto-summary
```

```
R2#show run | section eigrp
router eigrp 21
 network 192.168.12.0
 auto-summary
```

I can also take a quick look at the running configuration and I'll see the same thing.

```
R2(config)#no router eigrp 21

router eigrp 12
 network 192.168.12.0
 metric weights 0 1 1 1 1 0
 auto-summary
```

Let's change the AS number on R2.

```
R1# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2
(FastEthernet0/0) is up: new adjacency
```

```
R2# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.1
(FastEthernet0/0) is up: new adjacency
```

After changing the AS number life is good. **Lesson learned: Make sure the AS number is the same if you want an EIGRP neighbor adjacency.**

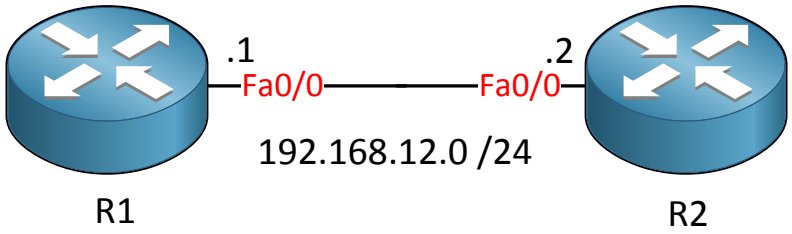
```
R1#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)       Cnt
Num
0   192.168.12.2            Fa0/0             11 00:01:44   13   200  0  3
```

```
R2#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)       Cnt
Num
0   192.168.12.1            Fa0/0             12 00:01:53    7   200  0  9
```

What if I only have access to one of the routers? That will be a challenge but it's possible to debug the AS number from the incoming EIGRP packets. If you want to see how this is done you can take a look at the following lab I created:

http://gns3vault.com/EIGRP/eigrp-debug-as-number.html

Last but not least…if you checked the AS number, K values, IP addresses and you still don't have a working EIGRP neighbor adjacency then you should think about security. Maybe an access-list is blocking EIGRP and/or multicast traffic.



Once again two EIGRP routers and no neighbor adjacency. What is going on?

```
R1#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
```

```
R2#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
```

We see that there are no neighbors…

```
R1#show ip protocols
Routing Protocol is "eigrp 12"
  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 12
  EIGRP NSF-aware route hold timer is 240s
  Automatic network summarization is in effect
  Maximum path: 4
  Routing for Networks:
    192.168.12.0
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: internal 90 external 170
```

```
R2#show ip protocols
Routing Protocol is "eigrp 12"
  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 12
  EIGRP NSF-aware route hold timer is 240s
  Automatic network summarization is in effect
  Maximum path: 4
  Routing for Networks:
    192.168.12.0
  Passive Interface(s):
    FastEthernet0/0
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: internal 90 external 170
```

If you look at the output of show ip protocols you can see that the network has been advertised correctly. If you look closely on R2 you can see that we have a passive interface,

Let's get rid of it!

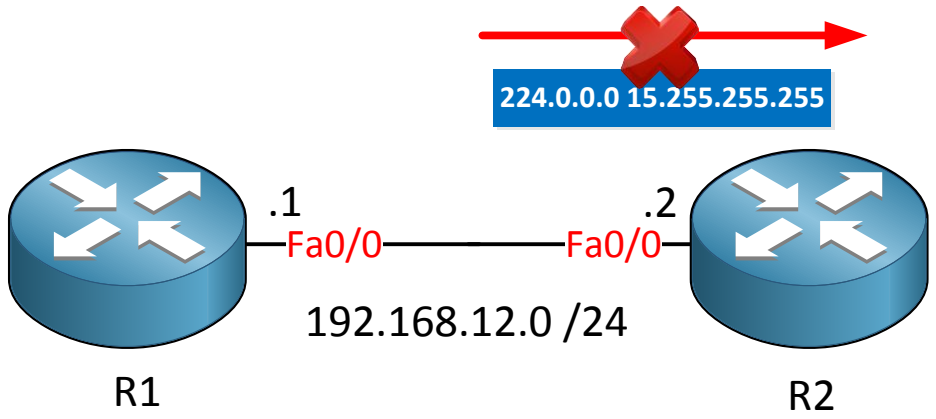
```
R2(config)#router eigrp 12
R2(config-router)#no passive-interface fastEthernet 0/0
```

Another misconfiguration bites the dust…

```
R1#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)       Cnt
Num
0   192.168.12.2            Fa0/0             13 00:05:23   24   200  0   6
```

```
R2#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)       Cnt
Num
0   192.168.12.1            Fa0/0             14 00:05:39   20   200  0   6
```

There we go! Problem solved! **Lesson learned: Don't enable passive interface if you want to establish an EIGRP neighbor adjacency.**



In the example above I have the same 2 routers but this time someone decided it was a good idea to configure an access-list on R2 that blocks all incoming multicast traffic.

```
R1# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2
(FastEthernet0/0) is down: retry limit exceeded
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2 (FastEthernet0/0)
is up: new adjacency
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2 (FastEthernet0/0)
is down: retry limit exceeded
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2 (FastEthernet0/0)
is up: new adjacency
```

This is where things might become confusing. On R1 you can see that it believes it has established an EIGRP neighbor adjacency with R2. This happens because we are still

receiving EIGRP packets from R2.

```
R1#debug eigrp neighbors
EIGRP Neighbors debugging is on
EIGRP: Retransmission retry limit exceeded
EIGRP: Holdtime expired
```

We can do a **debug eigrp neighbors** to see what is going on. Apparently R1 is not getting a response back from its hello messages, the holdtime expires and it will drop the EIGRP neighbor adjacency.

```
R1#ping 224.0.0.10

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 224.0.0.10, timeout is 2 seconds:
.
```

A quick way to test connectivity is to send a ping to the 224.0.0.10 multicast address that EIGRP uses. You can see we don't get a response from this ping. It's a good idea to check if there are access-lists in the network.

```
R2#show ip interface fa0/0 | include access list
  Outgoing access list is not set
  Inbound  access list is BLOCKMULTICAST
```

Bingo! We found something…

```
R2#show ip access-lists
Extended IP access list BLOCKMULTICAST
    10 deny ip any 224.0.0.0 15.255.255.255 (536 matches)
    20 permit ip any any (468 matches)
```

This access-list is blocking all multicast traffic. Let's punch a hole in it so EIGRP is allowed.

```
R2(config)#ip access-list extended BLOCKMULTICAST
R2(config-ext-nacl)#5 permit ip any host 224.0.0.10
```

We can create a specific statement that will allow EIGRP traffic.

```
R2#show access-lists
Extended IP access list BLOCKMULTICAST
    5 permit ip any host 224.0.0.10 (27 matches)
    10 deny ip any 224.0.0.0 15.255.255.255 (569 matches)
    20 permit ip any any (501 matches)
```

You can see our EIGRP traffic matches the statement I just created.

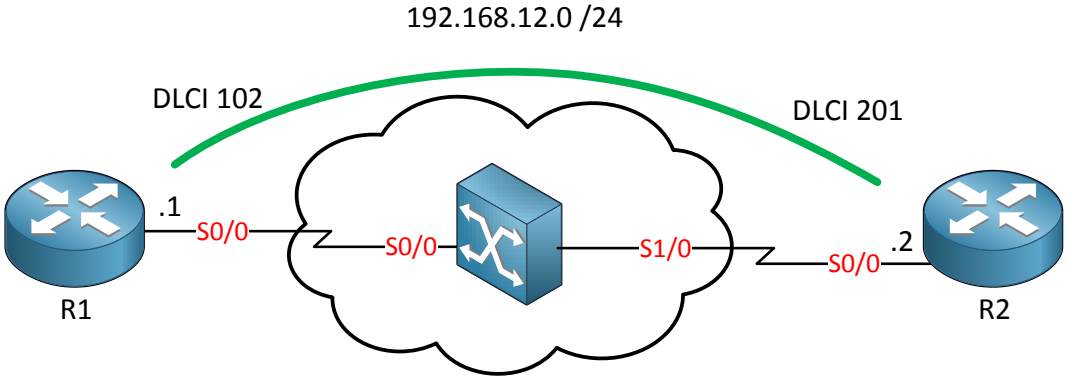
```
R1# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2
(FastEthernet0/0) is up: new adjacency
```

```
R2# %DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.1
(FastEthernet0/0) is up: new adjacency
```

Both routers now show a working EIGRP neighbor adjacency.

```
R1#ping 224.0.0.10

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 224.0.0.10, timeout is 2 seconds:

Reply to request 0 from 192.168.12.2, 24 ms
```

The ping that I just tried is now working. **Lesson learned: Don't block EIGRP packets!**



One more issue I want to share with you that can prevent EIGRP from becoming neighbors. In the picture above we have a frame-relay network and there's one PVC between R1 and R2. Here is the relevant configuration:

```
R1#
interface Serial0/0
 ip address 192.168.12.1 255.255.255.0
 encapsulation frame-relay
 serial restart-delay 0
 frame-relay map ip 192.168.12.2 102
 no frame-relay inverse-arp

router eigrp 12
 network 192.168.12.0
 auto-summary
```

```
R2#
interface Serial0/0
 ip address 192.168.12.2 255.255.255.0
 encapsulation frame-relay
 serial restart-delay 0
 frame-relay map ip 192.168.12.1 201
 no frame-relay inverse-arp

router eigrp 12
 network 192.168.12.0
 auto-summary
```

Both routers are configured for frame-relay and EIGRP has been configured.

```
R1#show ip eigrp neighbors
```

```
IP-EIGRP neighbors for process 12
```

```
R2#show ip eigrp neighbors
IP-EIGRP neighbors for process 12
```

You can see that there are no neighbors…that's not good! Can I ping the other side?

```
R1#ping 192.168.12.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/8/24 ms
```

Sending a ping is no problem so we can assume the frame-relay PVC is working. EIGRP however uses multicast and frame-relay by default is NBMA. Can we ping the 224.0.0.10 EIGRP multicast address?

```
R1#ping 224.0.0.10

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 224.0.0.10, timeout is 2 seconds:
.
```

No response here, at least we now know that unicast traffic is working and multicast doesn't work. Frame-relay can be configured for point-
to-point or point-to-multipoint. A physical interface is always a frame-relay point-to-multipoint interface and those require frame-relay maps, let's check it out:

```
R1#show frame-relay map
Serial0/0 (up): ip 192.168.12.2 dlci 102(0x66,0x1860), static,
              CISCO, status defined, active
```

```
R2#show frame-relay map
Serial0/0 (up): ip 192.168.12.1 dlci 201(0xC9,0x3090), static,
              CISCO, status defined, active
```

We can see both routers have a DLCI-to-IP mapping so they know how to reach each other. I can see they keyword **"static"** which also reveals to me that this mapping was configured by someone and not learned through Inverse ARP (otherwise you see "dynamic"). I don't see the **"broadcast"** keyword which is required to forward broadcast or multicast traffic. At this moment we have 2 options to fix this problem:

- Configure EIGRP to use unicast traffic instead of multicast.
- Check the frame-relay configuration and make sure multicast traffic can be forwarded.

Let's do the EIGRP unicast configuration first:

```
R1(config)#router eigrp 12
R1(config-router)#neighbor 192.168.12.2 serial 0/0
```

```
R2(config)#router eigrp 12
```

```
R2(config-router)#neighbor 192.168.12.1 serial 0/0
```

We require the **neighbor** command for the EIGRP configuration. As soon as you type in these commands and hit enter you'll see this:

```
R1#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2 (Serial0/0) is up:
new adjacency
```

```
R2#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.1 (Serial0/0) is up:
new adjacency
```

Problem solved! Now let's try the other method where we send multicast traffic down the frame-relay PVC:

```
R1(config)#router eigrp 12
R1(config-router)#no neighbor 192.168.12.2 serial 0/0
```

```
R2(config)#router eigrp 12
R2(config-router)#no neighbor 192.168.12.1 serial 0/0
```

If it ain't broke…don't fix it…not this time! Time to hammer down the EIGRP neighbor adjacency.

```
R1(config)#interface serial 0/0
R1(config-if)#frame-relay map ip 192.168.12.2 102 broadcast
```

```
R2(config)#interface serial 0/0
R2(config-if)#frame-relay map ip 192.168.12.1 201 broadcast
```

Broadcast is the magic keyword here. This will allow broadcast and multicast traffic.

```
R1#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.2 (Serial0/0) is up:
new adjacency
```

```
R2#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 12: Neighbor 192.168.12.1 (Serial0/0) is up:
new adjacency
```

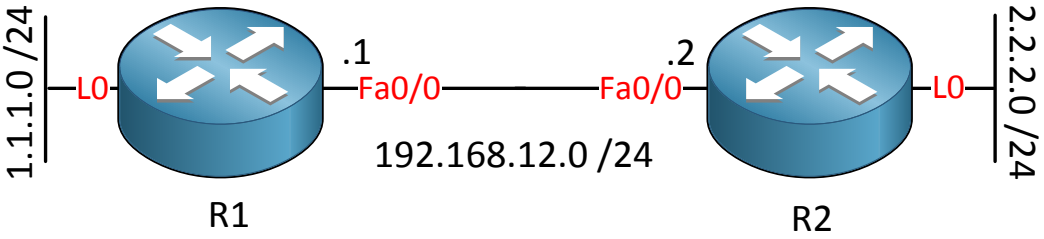
After changing the frame-relay map configuration we have an EIGRP neighbor adjacency! That's all there is to it. **Lesson learned: Check if your frame-relay network supports broadcast or not. Configure EIGRP to use unicast or change your frame-relay configuration to support broadcast traffic.**

Excellent! You have now seen the most common errors that can cause EIGRP not to form an EIGRP neighbor adjacency. If you want to get some practice right now you might want to try this lab that teaches you how to solve all the different EIGRP neighbor adjacency issues:

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

Now we can continue with troubleshooting route advertisements. Most of the time you are expecting to see a certain network in the routing table but it's not there. I'll show you a number of things that could go wrong with EIGRP and how to fix them, here are the most common errors:

- Someone configured a distribute-list so routing information is being filtered.
- Auto-summarization has been configured or someone created a manual summary.
- Split-horizon is preventing the advertisement of routing information.
- Redistribution has been configured but information from EIGRP is not being used.
- Redistribution has been configured but no EIGRP external routes are showing up.



Let's start with a simple topology. R1 and R2 are running EIGRP and each router has a loopback interface. Here's the configuration of both routers:

```
R1(config)#router eigrp 12
R1(config-router)#no auto-summary
R1(config-router)#network 1.1.1.0 0.0.0.255
R1(config-router)#network 192.168.12.0 0.0.0.255
```

```
R2(config)#router eigrp 12
R2(config-router)#no auto-summary
R2(config-router)#network 2.2.2.0 0.0.0.255
R2(config-router)#network 192.168.12.0 0.0.0.255
```

Everything is working fine until a couple of weeks later one of the users is complaining that they are unable to reach the 2.2.2.0 /24 network from behind R1. You take a look at the routing table on R1 and this is what you see:

```
R1#show ip 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
```

For some reason you don't see network 2.2.2.0 /24 in the routing table.

```
R1#show ip protocols | include filter
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
```

I can see no distribute lists have been configured on R1.

```
R2#show ip 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
D       1.1.1.0 [90/156160] via 192.168.12.1, 00:14:01, FastEthernet0/0
     2.0.0.0/24 is subnetted, 1 subnets
C       2.2.2.0 is directly connected, Loopback0
```

R2 does have network 1.1.1.0 /24 in its routing table. Let's do a quick debug to see what is going on.

```
R2#debug ip eigrp
IP-EIGRP Route Events debugging is on
IP-EIGRP(Default-IP-Routing-Table:12): 2.2.2.0/24 - denied by distribute
list
```

A debug quickly shows us what is going on. It make tight a while before you see this message or you can reset the EIGRP neighbor adjacency to speed things up. As you can see network 2.2.2.0 /24 is being denied because of a distribute list.

```
R2#show ip protocols | include filter
  Outgoing update filter list for all interfaces is 1
  Incoming update filter list for all interfaces is not set
```

Another quick method of checking this is using the **show ip protocols** command.

```
R2#show run | section eigrp
router eigrp 12
 network 2.2.2.0 0.0.0.255
 network 192.168.12.0
 distribute-list 1 out
 no auto-summary
```

Using show run might have been quicker to spot the distribute-list in this case.

```
R2#show access-lists
Standard IP access list 1
    10 deny   2.2.2.0, wildcard bits 0.0.0.255 (5 matches)
    20 permit any (5 matches)
```

Here's the access-list causing us trouble.

```
R2(config)#router eigrp 12
R2(config-router)#no distribute-list 1 out
```

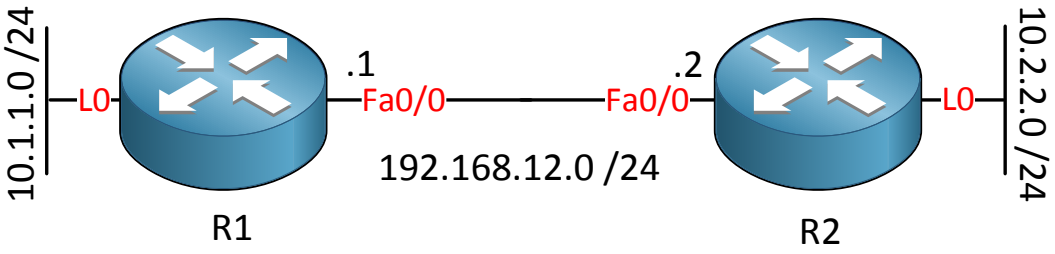
Let's get rid of the distribute-list.

```
R1#show ip 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, 00:00:13, FastEthernet0/0
```

Problem solved! **Lesson learned: If the network commands are correct, check if you have a distribute-list that is preventing prefixes from being advertised or installed in the routing table.**

Keep in mind distribute-lists can be configured **inbound** or **outbound** just like an access-list.



Onto the next scenario, same 2 routers but different networks on the loopbacks, here's the configuration:

```
R1(config)#router eigrp 12
R1(config-router)#network 192.168.12.0
R1(config-router)#network 10.0.0.0
```

```
R2(config)#router eigrp 12
R2(config-router)#network 192.168.12.0
R2(config-router)#network 10.0.0.0
```

It's a pretty basic configuration as you can see.

```
R1#show ip route

Gateway of last resort is not set

C    192.168.12.0/24 is directly connected, FastEthernet0/0
     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.1.1.0/24 is directly connected, Loopback0
D       10.0.0.0/8 is a summary, 00:00:07, Null0
```

```
R2#show ip route

Gateway of last resort is not set

C    192.168.12.0/24 is directly connected, FastEthernet0/0
     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.2.2.0/24 is directly connected, Loopback0
D       10.0.0.0/8 is a summary, 00:00:15, Null0
```

Looking at the routing tables, I don't see network 10.1.1.0 /24 or 10.2.2.0 /24. I do see an

entry for the 10.0.0.0/8 network pointing to the null0 interface. This entry only shows up when summarization is configured and it's used to prevent routing loops.

```
R2#debug ip eigrp
IP-EIGRP Route Events debugging is on
```

Let's enable a debug and see what we can find.

```
R2#clear ip eigrp 12 neighbors
```

I'll do a reset of the EIGRP neighbor adjacency to speed things up, keep in mind this is probably not the best thing to do on a production network until you know what's wrong but it does help to speed things up.

```
R2#
IP-EIGRP(Default-IP-Routing-Table:12): 192.168.12.0/24 - do advertise out
FastEthernet0/0
IP-EIGRP(Default-IP-Routing-Table:12): 10.2.2.0/24 - don't advertise out
FastEthernet0/0
IP-EIGRP(Default-IP-Routing-Table:12): 10.0.0.0/8 - do advertise out
FastEthernet0/0
```

Here's our answer. The debug tells us that 10.2.2.0 /24 should not be advertised but 10.0.0.0 /8 has to be advertised (a summary). This can happen because of 2 reasons:

- Summarization was configured by someone.
- Auto-summary is enabled for EIGRP.

```
R1#show run | section eigrp
router eigrp 12
 network 10.0.0.0
 network 192.168.12.0
 auto-summary
```

```
R2#show run | section eigrp
router eigrp 12
 network 10.0.0.0
 network 192.168.12.0
 auto-summary
```

As you can see **auto-summary** is enabled for EIGRP. Depending on the IOS version this is enabled or disabled by default.

```
R1(config)#router eigrp 12
R1(config-router)#no auto-summary
```

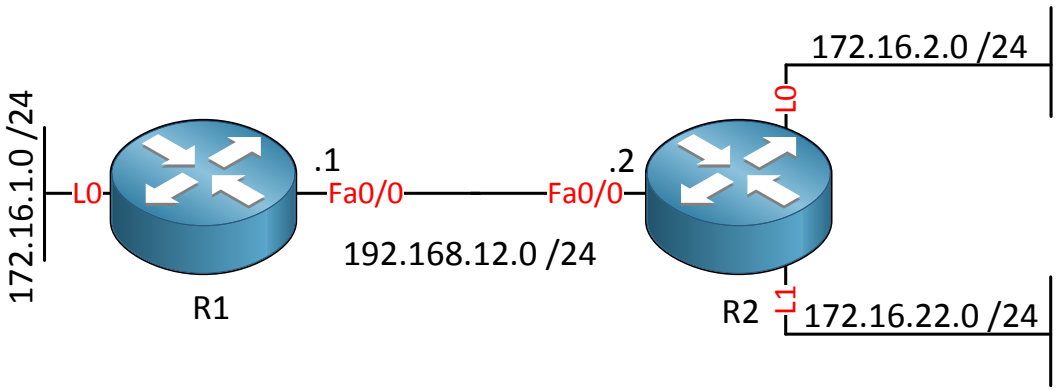
```
R2(config)#router eigrp 12
R2(config-router)#no auto-summary
```

Disabling auto-summarization should do the trick.

```
R1#show ip route

C    192.168.12.0/24 is directly connected, FastEthernet0/0
     10.0.0.0/24 is subnetted, 2 subnets
D       10.2.2.0 [90/156160] via 192.168.12.2, 00:00:22, FastEthernet0/0
C       10.1.1.0 is directly connected, Loopback0
```

```
R2#show ip route

C    192.168.12.0/24 is directly connected, FastEthernet0/0
     10.0.0.0/24 is subnetted, 2 subnets
C       10.2.2.0 is directly connected, Loopback0
D       10.1.1.0 [90/156160] via 192.168.12.1, 00:16:24, FastEthernet0/0
```

Now we see both networks appearing in the routing table. **Lesson learned: If EIGRP auto-summary is enabled you might end up with discontigous networks.**



Let me show you another issue that can arise with summarization. In the example above we have 2 routers but different networks. R1 has network 172.16.1.0 /24 on a loopback interface and R2 has network 172.16.2.0 /24 and 172.16.22.0 /24 on its loopback interfaces. Let me show you the EIGRP configuration of both routers:

```
R1#
router eigrp 12
 network 172.16.1.0 0.0.0.255
 network 192.168.12.0
 auto-summary
```

```
R2#
router eigrp 12
 network 172.16.2.0 0.0.0.255
 network 172.16.22.0 0.0.0.255
 network 192.168.12.0
 no auto-summary
```

You can see that all networks are advertised. Note that R1 has auto-summary enabled and R2 has auto-summary disabled.

```
R2#
interface FastEthernet0/0
ip summary-address eigrp 12 172.16.0.0 255.255.0.0 5
```

Someone configured a summary on R2 and is sending it towards R1. The summary that was created is 172.16.0.0 /16.

```
R1#show ip route

C    192.168.12.0/24 is directly connected, FastEthernet0/0
     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
D       172.16.0.0/16 is a summary, 00:30:07, Null0
C       172.16.1.0/24 is directly connected, Loopback0
```

However if I look at the routing table of R1 it doesn't show up. We do see an entry for the 172.16.0.0 /16 network but it's pointing to the null0 interface…not towards R2. What is going on here?

```
R2#debug ip eigrp
IP-EIGRP Route Events debugging is on
```

```
R2#clear ip eigrp 12 neighbors
```

Let's do a debug on R2 to see if the summary is being advertised. I'll do a clear ip eigrp neighbors too just to speed things up.

```
R2#
IP-EIGRP(Default-IP-Routing-Table:12): 192.168.12.0/24 - do advertise out
FastEthernet0/0
IP-EIGRP(Default-IP-Routing-Table:12): 172.16.2.0/24 - don't advertise out
FastEthernet0/0
IP-EIGRP(Default-IP-Routing-Table:12): 172.16.22.0/24 - don't advertise out
FastEthernet0/0
IP-EIGRP(Default-IP-Routing-Table:12): 172.16.0.0/16 - do advertise out
FastEthernet0/0
```

Looking at the debug we can see that R2 is working properly. It's advertising the 172.16.0.0 /16 summary route as it should. This means the problem has to be at R1.

```
R1#debug ip eigrp
IP-EIGRP Route Events debugging is on
```

Let's debug R1.

```
R1#
IP-EIGRP(Default-IP-Routing-Table:12): Processing incoming UPDATE packet
IP-EIGRP(Default-IP-Routing-Table:12): Int 172.16.0.0/16 M 156160 - 25600
130560 SM 128256 - 256 128000
IP-EIGRP(Default-IP-Routing-Table:12): route installed for 172.16.0.0
(Summary)
IP-EIGRP(Default-IP-Routing-Table:12): 172.16.0.0/16 routing table not
updated thru 192.168.12.2
```

We can see that R1 receives the summary route from R2 but decides not to use it.

```
R1#show ip eigrp topology 172.16.0.0
```

```
IP-EIGRP (AS 12): Topology entry for 172.16.0.0/16
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 128256
  Routing Descriptor Blocks:
  0.0.0.0 (Null0), from 0.0.0.0, Send flag is 0x0
      Composite metric is (128256/0), Route is Internal
      Vector metric:
        Minimum bandwidth is 10000000 Kbit
        Total delay is 5000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1514
        Hop count is 0
  192.168.12.2 (FastEthernet0/0), from 192.168.12.2, Send flag is 0x0
      Composite metric is (156160/128256), Route is Internal
      Vector metric:
        Minimum bandwidth is 100000 Kbit
        Total delay is 5100 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
```

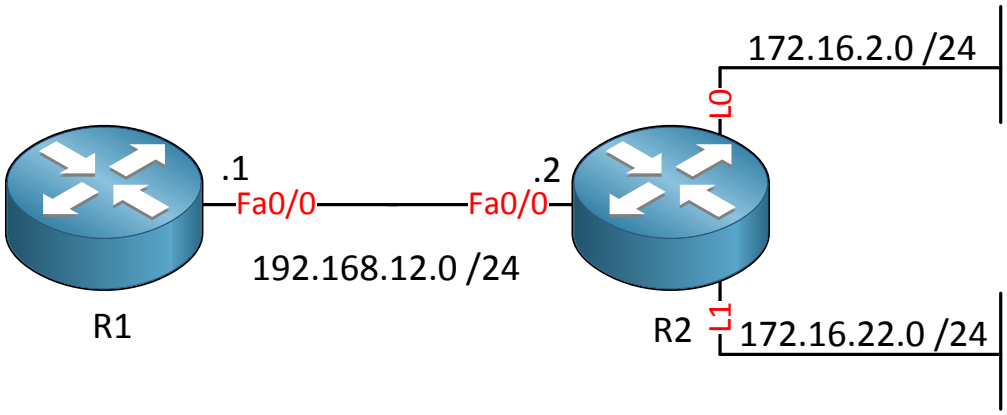
This is a good moment to check the EIGRP topology table. You can see that it does have the 172.16.0.0 /16 summary from R2 in its EIGRP topology table but R1 decides not to use it because the entry to the null0 interface is a better path.

```
R1(config)#router eigrp 12
R1(config-router)#no auto-summary
```

The solution is that we need to get rid of the null0 entry in the routing table. The only way to do this is by disabling auto summarization.

```
R1#show ip route

C    192.168.12.0/24 is directly connected, FastEthernet0/0
     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
D       172.16.0.0/16 [90/156160] via 192.168.12.2, 00:00:51,
FastEthernet0/0
C       172.16.1.0/24 is directly connected, Loopback0
```

There we go. Disabling auto summarization removes the null0 entry and now the summary of R2 can be installed…problem solved! **Lesson learned: EIGRP auto-summary creates an entry to the null0 interface which might prevent the installation of summaries you receive from neighbor routers.**



There is one more issue with summarization that I want to demonstrate. We are using the topology you see above and this is the EIGRP configuration of both routers.

```
R1#
router eigrp 12
 network 192.168.12.0
 no auto-summary
```

```
R2#
router eigrp 12
 network 172.16.2.0 0.0.0.255
 network 172.16.22.0 0.0.0.255
 network 192.168.12.0
 no auto-summary
```

All networks are advertised and auto summarization is disabled on both routers.

```
R2#
interface FastEthernet0/0
ip summary-address eigrp 12 172.16.0.0 255.255.0.0 5
```

A summary has been configured on R2 and should be advertised towards R1.

```
R1#show ip route

C    192.168.12.0/24 is directly connected, FastEthernet0/0
```

Unfortunately I'm not seeing anything on R1. Let's check R2 to see what is wrong.

```
R2#debug ip eigrp
IP-EIGRP Route Events debugging is on

R2#clear ip eigrp 12 neighbors
```

When it comes to troubleshooting networking not Google but Debug and show commands are your friends.

```
R2#
IP-EIGRP(Default-IP-Routing-Table:12): 192.168.12.0/24 - do advertise out
FastEthernet0/0
```

Hmm this is the only network that R2 is advertising.

```
R2#show ip route

C    192.168.12.0/24 is directly connected, FastEthernet0/0
```

One of the **golden rules of routing**: You can't advertise what you don't have. Apparently R2 only knows about network 192.168.12.0 /24.

```
R2#show ip interface brief
Interface              IP-Address      OK? Method Status
Protocol
FastEthernet0/0        192.168.12.2    YES manual up                    up
Loopback0              172.16.2.2      YES manual administratively down
down
Loopback1              172.16.22.22    YES manual administratively down
down
```

Uhoh…this looks like a Friday afternoon error! Someone left a shutdown command on the loopback interfaces.

```
R2(config)#interface loopback 0
R2(config-if)#no shutdown
R2(config)#interface loopback 1
R2(config-if)#no shutdown
```

Let's enable the interfaces.

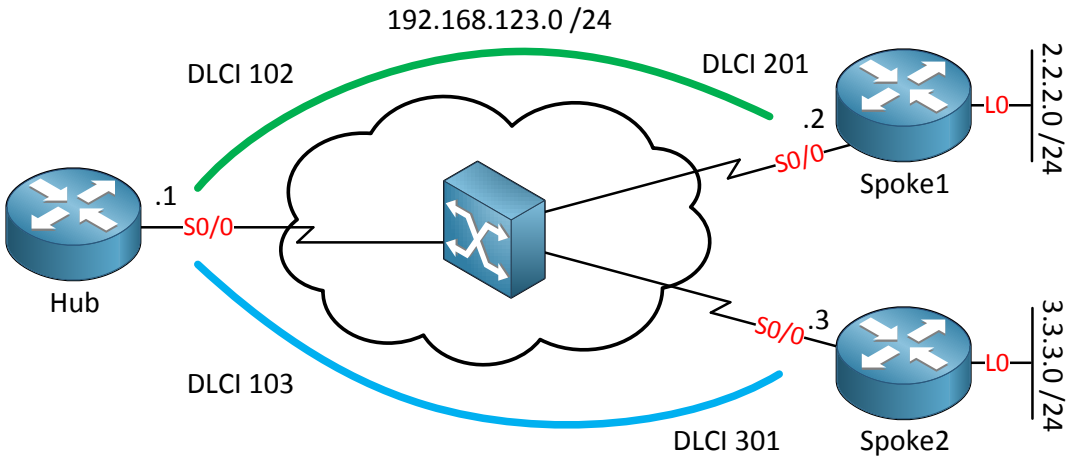
```
R2#
IP-EIGRP(Default-IP-Routing-Table:12): 172.16.2.0/24 - don't advertise out
FastEthernet0/0
IP-EIGRP(Default-IP-Routing-Table:12): 172.16.22.0/24 - don't advertise out
FastEthernet0/0
IP-EIGRP(Default-IP-Routing-Table:12): 172.16.0.0/16 - do advertise out
FastEthernet0/0
```

Now we see that the summary is being advertised.

```
R1#show ip route | include 172.16.0.0
D    172.16.0.0/16 [90/156160] via 192.168.12.2, 00:09:39, FastEthernet0/0
```

Now we see the summary in the routing table of R1, problem solved! **Lesson learned: You can't advertise wat you don't have in your routing table.**

The last problem might be simple but there's an important lesson you should not forget: In order for a summary route to be advertised at least one prefix that falls within the summary range has to be in the routing table of the advertising router!



Let's take a look at another topology. In the picture above we have a frame-relay hub and spoke topology. Spoke1 and Spoke2 each have a loopback interface which we will advertise in EIGRP. Here's the relevant configuration of all routers:

```
Hub(config)#router eigrp 123
Hub(config-router)#no auto-summary
Hub(config-router)#network 192.168.123.0
```

```
Spoke1(config-if)#router eigrp 123
Spoke1(config-router)#no auto-summary
Spoke1(config-router)#network 192.168.123.0
Spoke1(config-router)#network 2.2.2.0 0.0.0.255
```

```
Spoke2(config)#router eigrp 123
Spoke2(config-router)#no auto-summary
Spoke2(config-router)#network 192.168.123.0
Spoke2(config-router)#network 3.3.3.0 0.0.0.255
```

As you can see all networks are advertised.

```
Hub#show ip route

C    192.168.123.0/24 is directly connected, Serial0/0
     2.0.0.0/24 is subnetted, 1 subnets
D       2.2.2.0 [90/2297856] via 192.168.123.2, 00:00:11, Serial0/0
     3.0.0.0/24 is subnetted, 1 subnets
D       3.3.3.0 [90/2297856] via 192.168.123.3, 00:00:03, Serial0/0
```

Our hub router sees the networks of the 2 spoke routers.

```
Spoke1#show ip route

C    192.168.123.0/24 is directly connected, Serial0/0
     2.0.0.0/24 is subnetted, 1 subnets
C       2.2.2.0 is directly connected, Loopback0
```

```
Spoke2#show ip route

C    192.168.123.0/24 is directly connected, Serial0/0
     3.0.0.0/24 is subnetted, 1 subnets
C       3.3.3.0 is directly connected, Loopback0
```

Unfortunately our spoke routers don't see anything…

```
Hub#debug ip eigrp
IP-EIGRP Route Events debugging is on
```

It seems the Hub router doesn't advertise the networks it learns from the spoke routers. Let's enable a debug to see what is going on.

```
Hub#clear ip eigrp 123 neighbors
```

I'll reset the EIGRP neighbor adjacencies to speed things up.

```
R1# processing incoming UPDATE packet
IP-EIGRP(Default-IP-Routing-Table:123): Processing incoming UPDATE packet
IP-EIGRP(Default-IP-Routing-Table:123): Int 3.3.3.0/24 metric 2297856 -
1657856 640000
IP-EIGRP(Default-IP-Routing-Table:123): Processing incoming UPDATE packet
IP-EIGRP(Default-IP-Routing-Table:123): Int 2.2.2.0/24 M 2297856 - 1657856
640000 SM 128256 - 256 128000
IP-EIGRP(Default-IP-Routing-Table:123): route installed for 2.2.2.0  ()
IP-EIGRP(Default-IP-Routing-Table:123): 192.168.123.0/24 - do advertise out
Serial0/0
```

In the debug we can see that our Hub router learns about network 2.2.2.0 /24 and 3.3.3.0 /24 but only advertises network 192.168.123.0 /24 to the spoke routers. **Split horizon** is preventing the advertisements from one spoke router to another. (If you learn something on an interface…don't advertise it out of the same interface again).

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

Let's disable split horizon on the serial interface of the Hub router.

```
Hub#
IP-EIGRP(Default-IP-Routing-Table:123): 192.168.123.0/24 - do advertise out
Serial0/0
IP-EIGRP(Default-IP-Routing-Table:123): 3.3.3.0/24 - do advertise out
Serial0/0
IP-EIGRP(Default-IP-Routing-Table:123): 2.2.2.0/24 - do advertise out
Serial0/0
```

Now we can see that the Hub router does advertise all networks.

```
Spoke1#show ip route

C    192.168.123.0/24 is directly connected, Serial0/0
     2.0.0.0/24 is subnetted, 1 subnets
C       2.2.2.0 is directly connected, Loopback0
     3.0.0.0/24 is subnetted, 1 subnets
D       3.3.3.0 [90/2809856] via 192.168.123.1, 00:16:02, Serial0/0
```

```
Spoke2#show ip route

C    192.168.123.0/24 is directly connected, Serial0/0
     2.0.0.0/24 is subnetted, 1 subnets
D       2.2.2.0 [90/2809856] via 192.168.123.1, 00:16:33, Serial0/0
     3.0.0.0/24 is subnetted, 1 subnets
C       3.3.3.0 is directly connected, Loopback0
```

The spoke routers can now learn about each other's networks since split horizon has been disabled. This is looking good but we are not done yet. Lesson learned: RIP and EIGRP are distance vector routing protocols and use split horizon. **Split horizon prevents the advertisement of a prefix out of the interface where we learned it on.**

```
Spoke1#ping 3.3.3.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3.3.3.3, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

```
Spoke2#ping 2.2.2.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Even though the networks show up in the routing tables I'm unable to ping from one spoke router to another. This is not an EIGRP problem but it's related to frame-relay….we still have to fix it.

When Spoke1 sends an IP packet to Spoke2 the IP packet looks like this:

Let's think like a router for the moment and see what is happening here.

First we need to check if Spoke1 knows where to send 3.3.3.3 to:

```
Spoke1#show ip route 3.3.3.3
Routing entry for 3.3.3.0/24
  Known via "eigrp 123", distance 90, metric 2809856, type internal
  Redistributing via eigrp 123
  Last update from 192.168.123.1 on Serial0/0, 00:38:10 ago
  Routing Descriptor Blocks:
  * 192.168.123.1, from 192.168.123.1, 00:38:10 ago, via Serial0/0
      Route metric is 2809856, traffic share count is 1
      Total delay is 45000 microseconds, minimum bandwidth is 1544 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
```

There's an entry for 3.3.3.3 and the next hop IP address is 192.168.123.1 (Hub router). Are we able to reach 192.168.123.1?

```
Spoke1#ping 192.168.123.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.123.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/8 ms
```

No problem at all, it seems Spoke1 is able to forward packets meant for network 3.3.3.0 /24. Let's go to the Hub router.

```
Hub#ping 3.3.3.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3.3.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

The Hub router has no issues sending traffic to network 3.3.3.0 /24 so at this moment we can draw the conclusion that the problem must be at router Spoke2.

This is the IP packet that router Spoke2 receives, when it responds it will create a new IP packet that looks like this:

Is spoke2 able to reach IP address 192.168.123.2? Time to find out!

```
Spoke2#ping 192.168.123.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.123.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Now we know the problem…Spoke2 is unable to reach IP address 192.168.123.2.

```
Spoke2#show ip route | include 192.168.123.0
C    192.168.123.0/24 is directly connected, Serial0/0
```

If we look at the routing table of Spoke2 we can see that network 192.168.123.0 /24 is directly connected. From a layer 3 perspective we don't have any issues. Time to move down the OSI model and check layer 2…or maybe in between layer 2 and 3.

```
Spoke2#show frame-relay map
Serial0/0 (up): ip 192.168.123.1 dlci 301(0x12D,0x48D0), dynamic,
              broadcast,, status defined, active
```

Frame-relay uses Inverse ARP to bind layer 2 (DLCI) to layer 3 (IP address). You can see that there is no mapping for IP address 192.168.123.2.

```
Spoke2(config)#int s0/0
Spoke2(config-if)#frame-relay map ip 192.168.123.2 301
```

Let's add the frame-relay map ourselves.

```
Spoke2#show frame-relay map
Serial0/0 (up): ip 192.168.123.1 dlci 301(0x12D,0x48D0), dynamic,
              broadcast,, status defined, active
Serial0/0 (up): ip 192.168.123.2 dlci 301(0x12D,0x48D0), static,
              CISCO, status defined, active
```

Now router Spoke2 knows how to reach router Spoke1.

```
Spoke1#ping 3.3.3.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3.3.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/10/16 ms
```

Finally router Spoke1 is able to ping router Spoke2's loopback interface. When we try to ping from router Spoke2 to router Spoke1's loopback interface we'll have the same issue so we'll add a frame-relay map statement there as well:

```
Spoke1(config)#int s0/0
Spoke1(config-if)#frame-relay map ip 192.168.123.3 201
```

```
Spoke1#show frame-relay map
Serial0/0 (up): ip 192.168.123.1 dlci 201(0xC9,0x3090), dynamic,
              broadcast,, status defined, active
Serial0/0 (up): ip 192.168.123.3 dlci 201(0xC9,0x3090), static,
              CISCO, status defined, active
```

Now we have an extra frame-relay mapping on router Spoke1.

```
Spoke2#ping 2.2.2.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/8/8 ms
```

And our ping is successful! **Lesson learned: Make sure the next hop IP address is reachable and if needed add additional frame-relay map statements.**

We just got a little sidetracked from EIGRP to solving frame-relay problems but that's what network engineers do…sometimes troubleshooting is a deadly cocktail of multiple protocols. Another method of dealing with split-horizon is changing the point-to-multipoint configuration to a point-to-point configuration.

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

Click on the link below to get the full version.

Get How to Master CCNP TSHOOT Today