how to master CCNP SWITCH

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://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. Lab Equipment

Before we are going to start on our switching journey we are going to take a look at the lab equipment you will need. GNS3 is a very useful tool but it only supports the emulation of routers. You are unable to emulate a switch in GNS3 like a Cisco Catalyst 2950, 2960, 3550, 3560 or 3750.

Courtesy of Cisco Systems, Inc. Unauthorized use not permitted.

The closest you can get to emulate a switch in GNS3 is inserting this NM16-ESW Etherswitch module in your virtual router.

It adds 16 switch ports to your virtual router and supports basic trunking and spanning-tree features. Unfortunately this module is very limited and it doesn't cut it for CCNP SWITCH labs.

So what do we need? My advice is to buy some **real physical switches**. Don't be scared…I'm not going to advise you to buy ultra-high tech brand new switches! We are going to buy used Cisco switches that are easy to find and they won't burn a hole in your wallet…

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

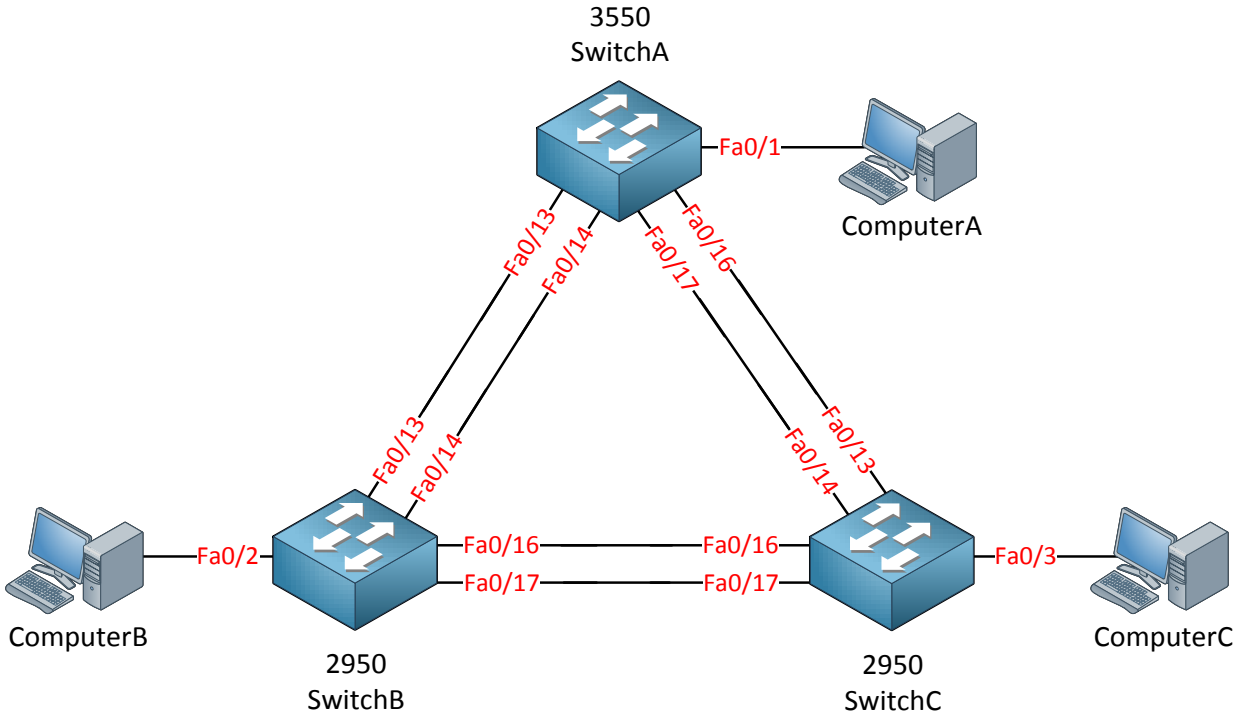
Without further ado…here are our candidates:

Courtesy of Cisco Systems, Inc. Unauthorized use not permitted.

Cisco Catalyst 2950: This is a layer 2 switch that can do all the vlan, trunking and spanning-tree stuff we need for CCNP SWITCH.

Cisco Catalyst 3550: This is a layer 3 switch. It offers pretty much the same features as the 2950 but it also supports routing.

If you look at eBay you can find the Cisco Catalyst 2950 for around $50, the Cisco Catalyst 3550 is around $100. It doesn't matter if you buy the 8, 24 or 48 port model. Not too bad right? Keep in mind you can sell them once you are done with CCNP without losing (much) money.



This is the topology I will be using throughout (most of) the book and I advise you to build it so you can do all the labs in this book by yourself. I did my best so you don't have to re-cable that often. We need one Cisco Catalyst 3550 because it can do routing; the other two Cisco Catalyst 2950 switches are sufficient for all the other stuff.

What about other switch models? Anything else we can use? Sure!

- The Cisco Catalyst 2960 is the successor of the Cisco Catalyst 2950, it's a great layer 2 switch but more expensive.
- The Cisco Catalyst 3560 is the successor of the Cisco Catalyst 3550, it also offers layer 3 features and it's quite more expensive…around $300 on eBay.
- The Cisco Catalyst 3750 is a layer 3 switch that is suitable for CCNP SWITCH.

I don't recommend buying the Cisco Catalyst 2960 because it doesn't offer anything extra compared to the Cisco Catalyst 2950 that'll help you beat the exam.

The Cisco Catalyst 3560 does offer two features that might justify buying it:

- It can do **private vlans** which is a CCNP SWITCH topic. It's impossible to configure it on a Cisco Catalyst 3550! It's a small topic though and personally I don't think it's worth the additional $200 just to configure private vlans.
- **QoS (Quality of Service)** is different on the Cisco Catalyst 3560 compared to the Cisco Catalyst 3550. If you intend to study QoS in the future I would recommend buying this switch. You won't need it for the CCNP SWITCH exam.

Are there any switches that you should NOT buy?

- Don't buy the Cisco Catalyst 2900XL switch; you'll need at least the Cisco Catalyst 2950 switch. Many features are not supported on the Cisco Catalyst 2900XL switch.
- Don't buy the Cisco Catalyst 3500XL switch, same problem as the one above.

If you studied CCNA you probably know the difference between straight-through and crossover cables. Modern switches and network cards support auto-sensing so it really doesn't matter what kind of cable you use.

If you are going to connect these older switches to each other make sure you **buy crossover cables** since they don't support auto-sensing!

I also like to use one of these. It's a USB connector with 4x RS-232 serial connectors you can use for your blue Cisco console cables to connect to your switches.

It saves the hassle of plugging and unplugging your console cable between your switches.

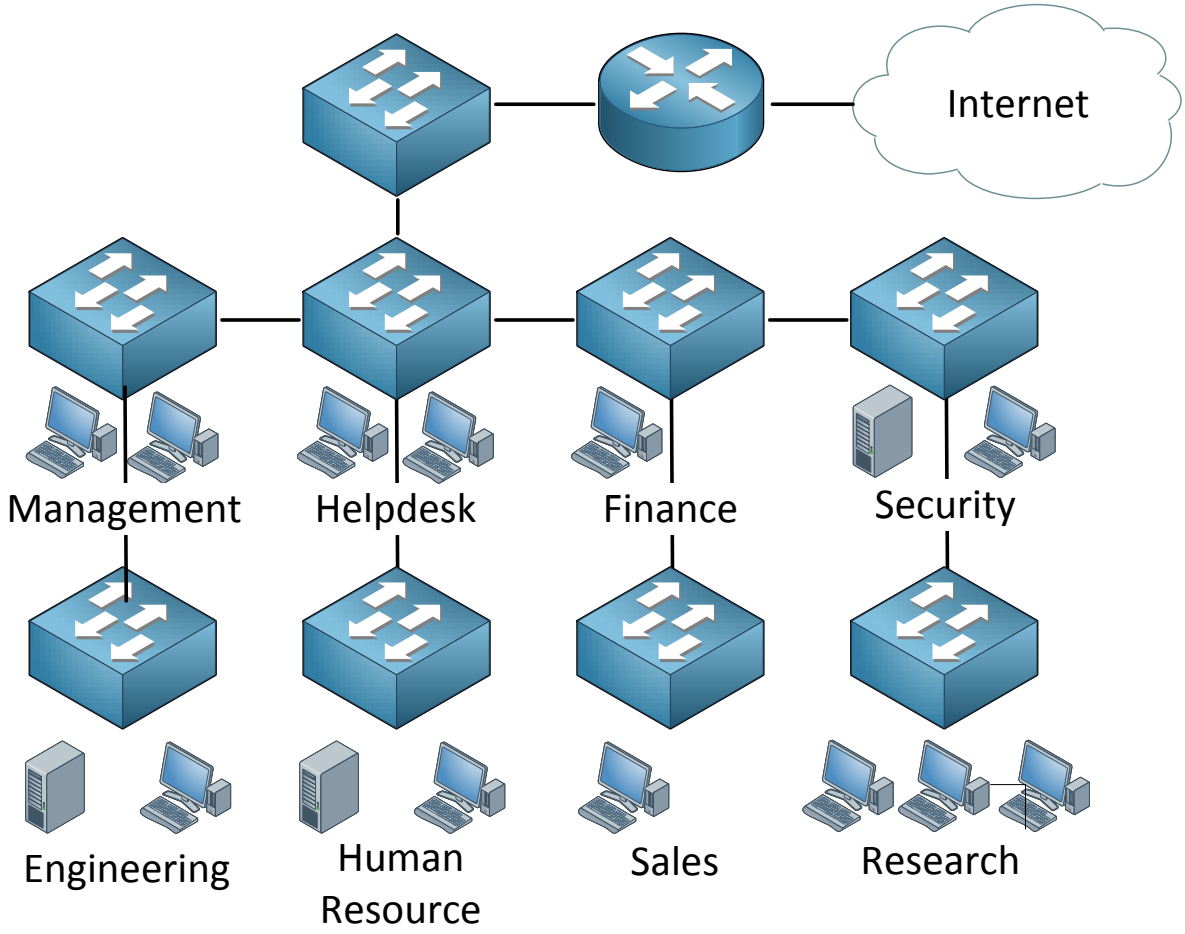
The one I'm using is from KÖNIG and costs around $30. Google for "USB 4x RS-232" and you should be able to find something similar.

In my topology picture you saw that I have three computers connected to my switches. For most of the labs I'm only using those computers to generate some traffic or send some pings so don't worry if you only have one computer, you can also use a cisco router if you have one.

2. VLANs (Virtual LANs)

In this chapter we will take a look at the configuration of VLANs, Trunks, Etherchannels and Private VLANs. If you studied CCNA then the first part of this chapter should be familiar to you.

Let's start off by looking at a picture of a network:



Look at this picture for a minute, we have many departments and each department has its own switch. Users are grouped physically together and are connected to their switch. What do you think of it? Does this look like a good network design? If you are unsure let me ask you some questions to think about:

- What happens when a computer connected to the Research switch sends a broadcast like an ARP request?
- What happens when the Helpdesk switch fails?
- Will our users at the Human Resource switch have fast network connectivity?
- How can we implement security in this network?

Now let me explain why this is a bad network design. If any of our computers sends a broadcast what will our switches do? They flood it! This means that a single broadcast frame will be flooded on this entire network. This also happens when a switch hasn't learned about a certain MAC address, the frame will be flooded.

If our helpdesk switch would fail this means that users from Human Resource are "isolated" from the rest and unable to access other departments or the internet, this applies to other switches as well. Everyone has to go through the Helpdesk switch in order to reach the Internet which means we are sharing bandwidth, probably not a very good idea performance-wise.

Last but not least, what about security? We could implement port-security and filter on MAC addresses but that's not a very secure method since MAC addresses are very easy to spoof. VLANs are one way to solve our problems.

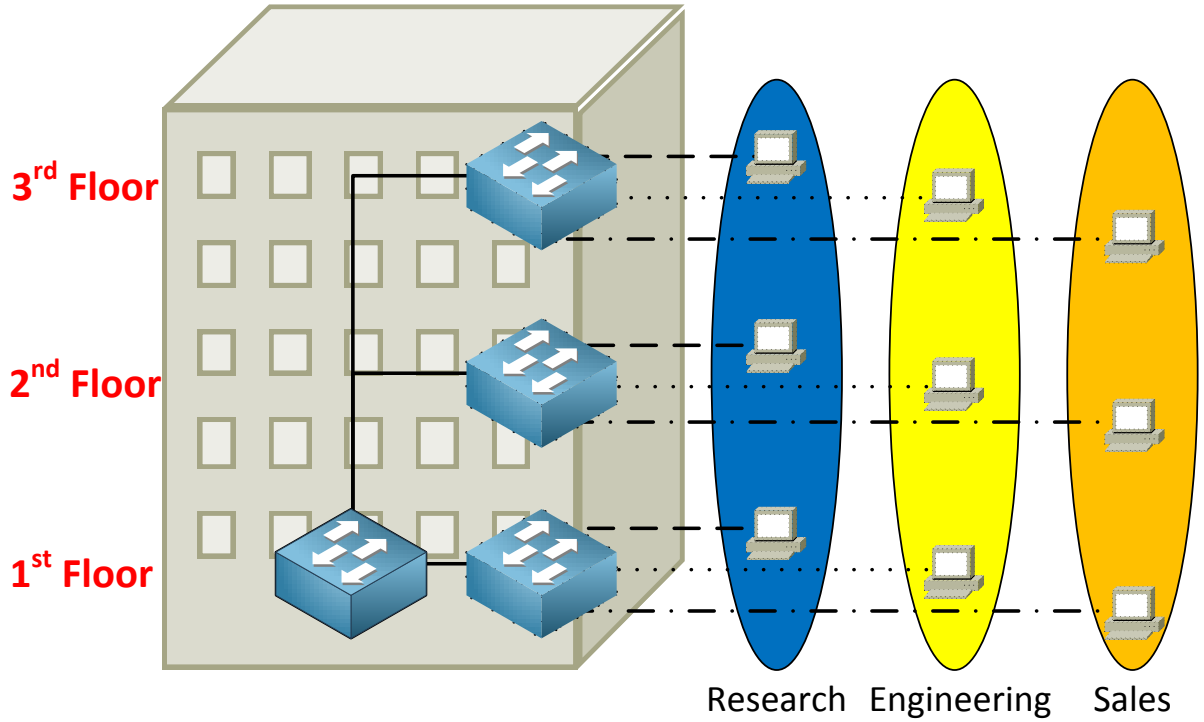
Two more questions I'd like to ask you to refresh your knowledge:

- How many collision domains do we have here?
- How many broadcast domains do we have here?

Each port on a switch is a separate collision domain so in this picture we have a LOT of collision domains…more than 20.

What about broadcast domains? If a computer from the Sales switch would send a broadcast frame we know that all other switches will forward it.

Routers don't forward broadcast frames so they effectively "limit" our broadcast domain. Of course on the right side of our router where we have an Internet connection this would be another broadcast domain…so we have 2 broadcast domains here.

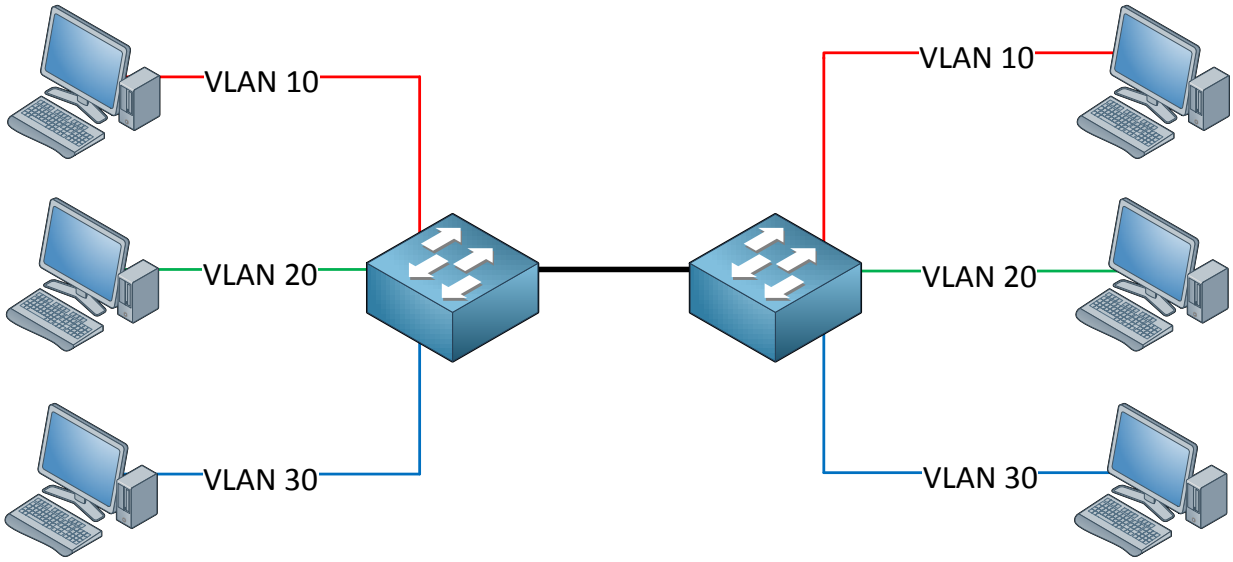


When you work with switches you have to keep in mind there's a big difference between physical and logical topology. Physical is just the way our cables are connected while logical is how we have configure things 'virtually'. In the example above we have 4 switches and I have created 3 VLANs called Research, Engineering and Sales. A VLAN is a Virtual LAN so it's like having a "switch inside a switch".

What are the advantages of using vlans?

- A VLAN is a single broadcast domain which means that if a user in the research VLAN sends a broadcast frame only users in the same VLAN will receive it.
- Users are only able to communicate within the same VLAN (unless you use a router).
- Users don't have to be grouped physically together, as you can see we have users in the Engineering vlan sitting on the 1st, 2nd and 3rd floor.

In my example I grouped different users in different VLANs but you can also use VLANs to separate different traffic types. Perhaps you want to have all printers in one VLAN, all servers in a VLAN and all the computers in another. What about VoIP? Put all your Voice over IP phones in a separate Vlan so its traffic is separated from other data (more on VoIP later!)



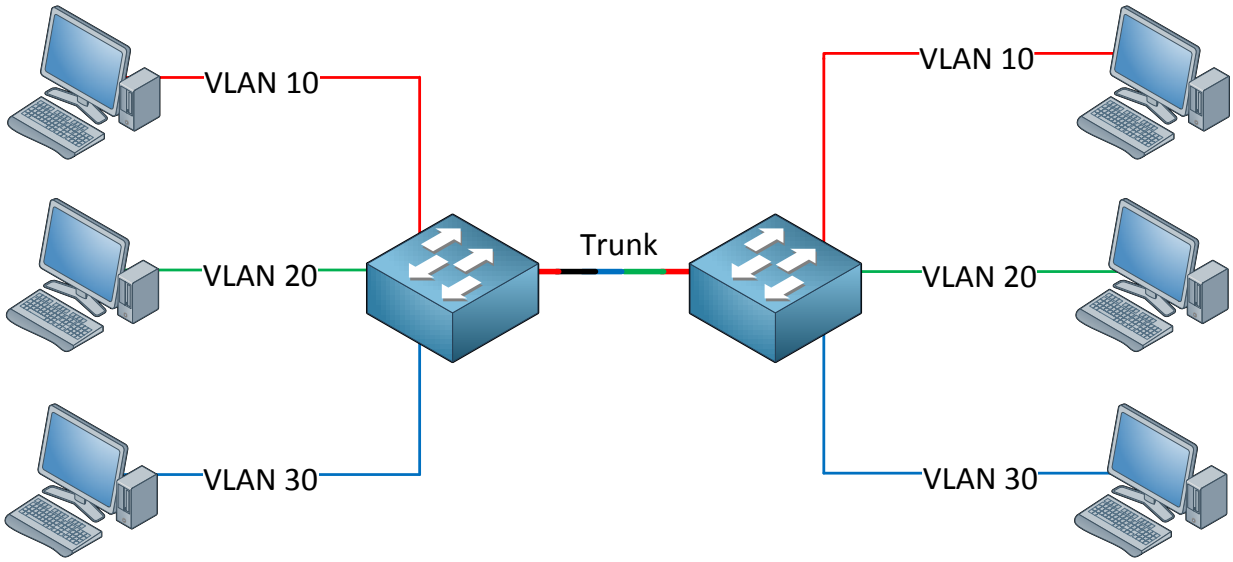
Let's take a look at the example above. There are three computers on each side belonging to three different VLANs. VLAN 10,20 and 30. There are two switches connecting these computers to each other.

Our switches will forward traffic but how do they know to which vlan our traffic belongs?

Let's take a look at an Ethernet frame:

Do you see any field where we can specify to which vlan our Ethernet frame belongs? Well there isn't! That's why we need a **trunking protocol** to help us.

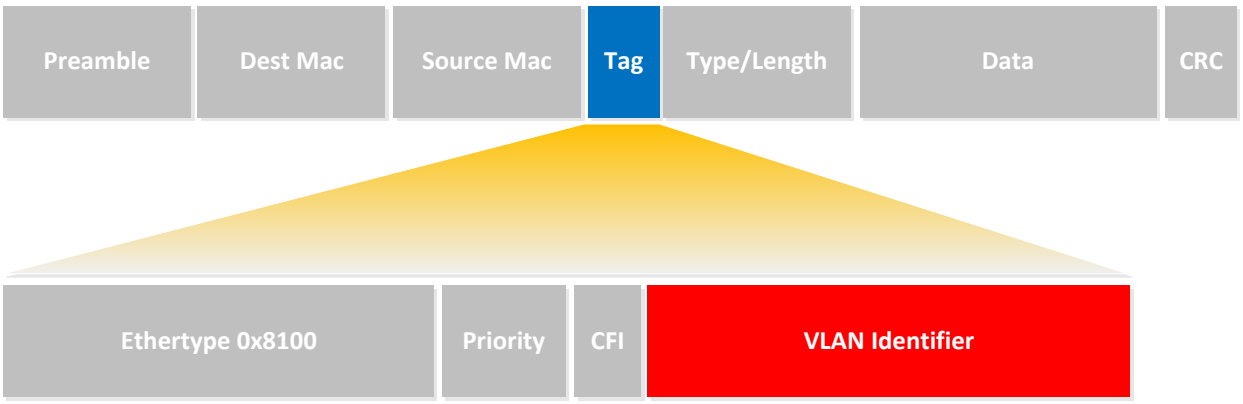
Between switches we are going to create a **trunk.** A trunk connection is simply said an interface that carries multiple VLANs.



There are **two trunking protocols** we can use:

- **IEEE 802.1Q**: An open standard that is supported on switches from many vendors and most NICs.
- **Cisco ISL (Inter-Switch Link)**: An old Cisco proprietary protocol that is only supported on some Cisco switches. If you bought some old Cisco catalyst 2950 switches you'll notice they only support 802.1Q.

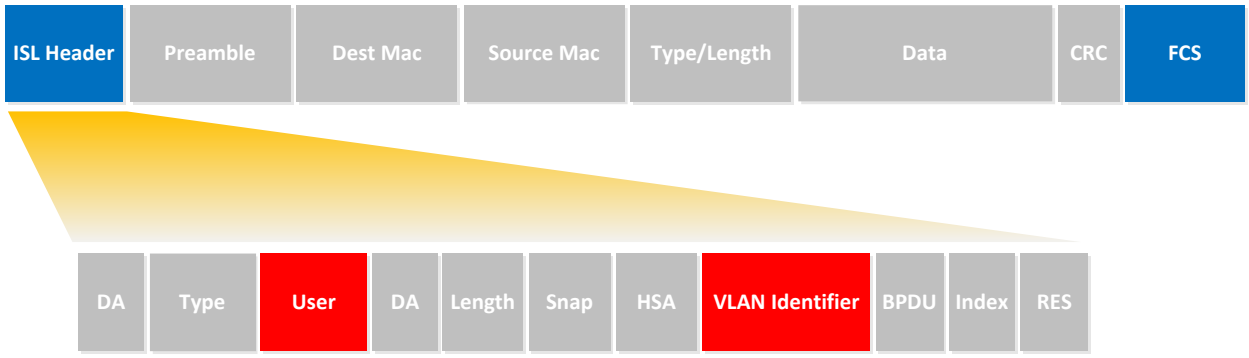
802.1Q FRAME



Let's start by looking at 802.1Q. In the picture you see an example of an 802.1Q Ethernet frame. As you can see it's the same as a normal Ethernet frame but we have added a **tag** in the middle (that's the blue field). In our tag you will find a "**VLAN identifier**" which is the VLAN to which this Ethernet frame belongs.
This is how switches know to which VLAN our traffic belongs. There's also a field called "**Priority**" which is used for QoS (Quality of Service). Keep in mind 802.1Q is a **standard**

and supported on switches from many different vendors. You can also use 802.1Q on many NICs.

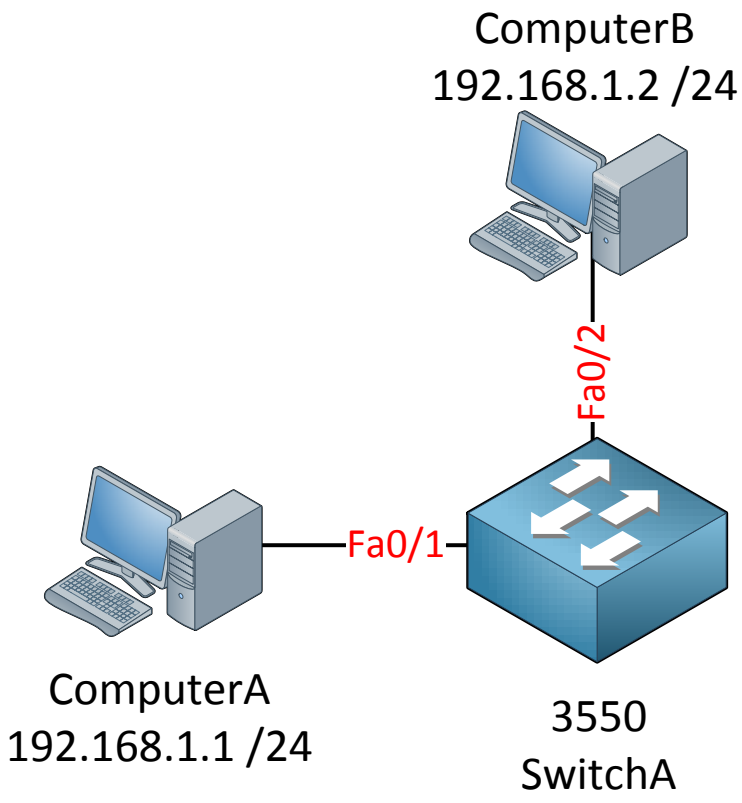
ISL FRAME



This is an example of an ISL Frame. The difference between 802.1Q and ISL is that 802.1 **tags** the Ethernet frame while ISL **encapsulates** the Ethernet Frame. You can see in the picture that ISL adds a new header in front of the Ethernet Frame and it adds a FCS (Frame Check Sequence). The header contains the "VLAN identifier" so we know to which VLAN this Ethernet Frame belongs. The **user** field is used for QoS (Quality of Service).

*If you studied CCNA you might recall the "native VLAN". On a Cisco switch this is VLAN 1 by default. The difference between 802.1Q and ISL concerning the native VLAN is that 802.1Q will **not tag** the native VLAN while ISL **does tag** the native VLAN.*

Enough theory for now, let's take a look at the configuration of VLANs and trunks.



Let's start with a simple example. ComputerA and ComputerB are connected to SwitchA.

First we will look at the default VLAN configuration on SwitchA:

```
SwitchA#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
1    default                           active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                 Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                 Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                 Fa0/13, Fa0/14, Fa0/15,
Fa0/22
                                                 Fa0/23, Fa0/24, Gi0/1, Gi0/2
1002 fddi-default                      act/unsup
1003 token-ring-default                act/unsup
1004 fddinet-default                   act/unsup
1005 trnet-default                     act/unsup
```

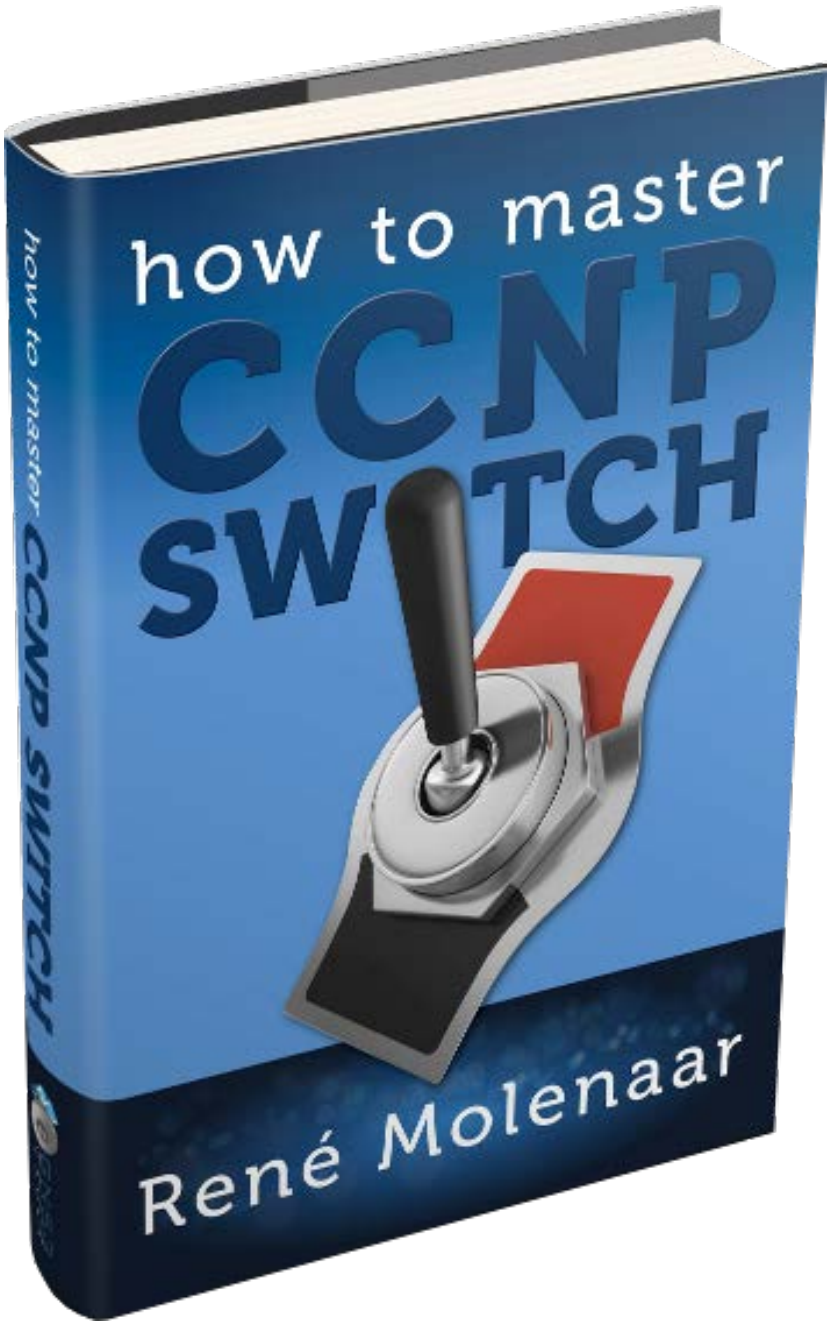
Interesting…VLAN 1 is the default VLAN and you can see that all interfaces are parked in VLAN 1.

*VLAN information is not saved in the running-config or startup-config but in a separate file called **vlan.dat** on your flash memory. If you want to delete the VLAN information you should delete this file by typing **delete flash:vlan.dat**.*

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

Click on the link below to get the full version.

Get How to Master CCNP SWITCH Today



I configured an IP address on ComputerA and ComputerB so they are in the same subnet.

```
C:\Documents and Settings\ComputerA>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
```

Even with the default switch configuration ComputerA is able to reach ComputerB. Let's see if I can create a new VLAN for ComputerA and ComputerB:

```
SwitchA(config)#vlan 50
SwitchA(config-vlan)#name Computers
SwitchA(config-vlan)#exit
```

This is how you create a new VLAN. If you want you can give it a name but this is optional. I'm calling my VLAN "Computers".

```
SwitchA#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
1    default                           active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                 Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                 Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                 Fa0/13, Fa0/14, Fa0/15,
                                                 Fa0/23, Fa0/24, Gi0/1, Gi0/2
50   Computers                         active
```

VLAN 50 was created on SwitchA and you can see that it's active. However no ports are currently in VLAN 50. Let's see if we can change this…

```
SwitchA(config)interface fa0/1
SwitchA(config-if)#switchport mode access
SwitchA(config-if)#switchport access vlan 50
```

```
SwitchA(config)interface fa0/2
SwitchA(config-if)#switchport mode access
SwitchA(config-if)#switchport access vlan 50
```

First I will configure the switchport in **access mode** with the **"switchport mode access" command.** By using the "**switchport access vlan**" command we can move our interfaces to another VLAN.

```
SwitchA#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--
1    default                          active    Fa0/3, Fa0/4
                                                Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/13, Fa0/14, Fa0/15,
                                                Fa0/23, Fa0/24, Gi0/1, Gi0/2
50   Computers                        active    Fa0/1, Fa0/2
```

Excellent! Both computers are now in VLAN 50. Let's verify our configuration by checking if they can ping each other:

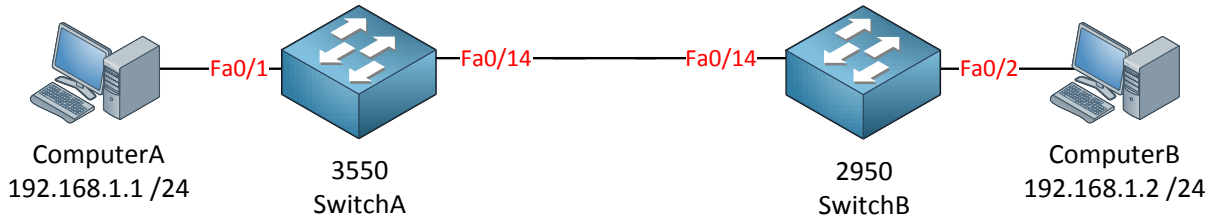
```
C:\Documents and Settings\ComputerA>ping 192.168.1.2

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
```

Our computers are able to reach each other within VLAN 50. Besides pinging each other we can also use another show command to verify our configuration:

```
SwitchA#show interfaces fa0/1 switchport
Name: Fa0/1
Switchport: Enabled
Administrative Mode: static access
Operational Mode: static access
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 50 (Computers)
Trunking Native Mode VLAN: 1 (default)
```

```
SwitchA#show interfaces fa0/2 switchport
Name: Fa0/1
Switchport: Enabled
Administrative Mode: static access
Operational Mode: static access
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 50 (Computers)
Trunking Native Mode VLAN: 1 (default)
```

By using the "show interfaces switchport" command we can see that the **operational mode** is "static access" which means it's in access mode. We can also verify that the interface is assigned to VLAN 50.



Let's continue our VLAN adventure by adding SwitchB to the topology. I also moved ComputerB from SwitchA to SwitchB.

```
SwitchB(config)#vlan 50
SwitchB(config-vlan)#name Computers
SwitchB(config-vlan)#exit
```

```
SwitchB(config)#interface fa0/2
SwitchB(config-if)#switchport access vlan 50
```

I just created VLAN 50 on SwitchB and the interface connected to ComputerB is assigned to VLAN 50.

Next step is to create a trunk between SwitchA and SwitchB:

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#switchport mode trunk
Command rejected: An interface whose trunk encapsulation is "Auto" can not be
configured to "trunk" mode.
```

```
SwitchB(config)#interface fa0/14
SwitchB(config-if)#switchport mode trunk
Command rejected: An interface whose trunk encapsulation is "Auto" can not be
configured to "trunk" mode.
```

I try to change the interface to trunk mode with the **"switchport mode trunk"** command. Depending on the switch model you might see the same error as me. If we want to change the interface to trunk mode we need to change the trunk encapsulation type. Let's see what options we have:

```
SwitchA(config-if)#switchport trunk encapsulation ?
  dot1q      Interface uses only 802.1q trunking encapsulation when trunking
  isl        Interface uses only ISL trunking encapsulation when trunking
  negotiate  Device will negotiate trunking encapsulation with peer on
             interface
```

Aha…so this is where you can choose between 802.1Q and ISL.

By default our switch will negotiate about the trunk encapsulation type.

```
SwitchA(config-if)#switchport trunk encapsulation dot1q
```

```
SwitchB(config-if)#switchport trunk encapsulation dot1q
```

Let's change it to 802.1Q by using the **"switchport trunk encapsulation"** command.

```
SwitchA#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
Administrative Trunking Encapsulation: dot1q
```

```
SwitchB#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
Administrative Trunking Encapsulation: dot1q
```

As you can see the trunk encapsulation is now 802.1Q.

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

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

Now I can successfully change the switchport mode to trunk.

```
SwitchA#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
```

```
SwitchB#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
```

We can confirm we have a trunk because the operational mode is "dot1q".

Let's try if ComputerA and ComputerB can reach each other:

```
C:\Documents and Settings\ComputerA>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
```

Excellent! ComputerA and ComputerB can reach each other! Does this mean we are done? Not quite yet…there's more I want to show to you:

```
SwitchB#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
1    default                           active    Fa0/1, Fa0/3, Fa0/4, Fa0/5
                                                 Fa0/6, Fa0/7, Fa0/8, Fa0/9
                                                 Fa0/10, Fa0/11, Fa0/12,
Fa0/13
                                                 Fa0/15, Fa0/22, Fa0/23,
Fa0/24
                                                 Gi0/1, Gi0/2
50   Computers                         active    Fa0/2
```

First of all, if we use the show vlan command we don't see the Fa0/14 interface. This is completely normal because the show vlan command **only shows interfaces in access mode** and **no trunk interfaces**.

```
SwitchB#show interface fa0/14 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/14      on               802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/14      1-4094

Port        Vlans allowed and active in management domain
Fa0/14      1,50

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/14      50
```

The **show interface trunk** is very useful. You can see if an interface is in trunk mode, which trunk encapsulation protocol it is using (802.1Q or ISL) and what the native VLAN is. We can also see that VLAN 1 – 4094 are allowed on this trunk.

We can also see that currently only VLAN 1 (native VLAN) and VLAN 50 are active. Last but not least you can see something which VLANs are in the forwarding state for spanning-tree (more on spanning-tree later!).

```
SwitchB(config-if)#switchport trunk allowed vlan ?
  WORD    VLAN IDs of the allowed VLANs when this port is in trunking mode
  add     add VLANs to the current list
  all     all VLANs
  except  all VLANs except the following
  none    no VLANs
  remove  remove VLANs from the current list
```

For security reasons it might be a good idea not to allow all VLANs on your trunk link. We can change this by using the **switchport trunk allowed vlan** command.

```
SwitchB(config-if)#switchport trunk allowed vlan remove 1-4094
SwitchB(config-if)#switchport trunk allowed vlan add 1-50
```

I just removed all allowed VLANs from the trunk and now only VLAN 1 – 50 are allowed.

```
SwitchB#show interface fa0/14 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/14      on               802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/14      1-50
```

Verify this by using the show interface trunk command.

```
SwitchB#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/14      on               802.1q         trunking      1
Fa0/16      auto             n-isl          trunking      1

Port        Vlans allowed on trunk
Fa0/14      1-50
Fa0/16      1-4094

Port        Vlans allowed and active in management domain
Fa0/14      1,50
Fa0/16      1,50

Port        Vlans allowed and active in management domain
Fa0/20      1,50
Fa0/21      1,50

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/14      50
Fa0/16      50
```

You can also use the **show interfaces trunk** command to get an overview of all your trunk interfaces. Besides our Fa0/14 interface you can see I got a couple of other interfaces that are in trunk mode.

Besides "access" and "trunk" mode we also have two "dynamic" methods. Let me show you what I mean:

```
SwitchB#show interface fa0/2 switchport
Name: Fa0/2
Switchport: Enabled
Administrative Mode: static access
Operational Mode: static access
```

An interface can be in access mode or in trunk mode. The interface above is connected to ComputerB and you can see that the operational mode is "static access" which means it's in access mode.

```
SwitchB#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
```

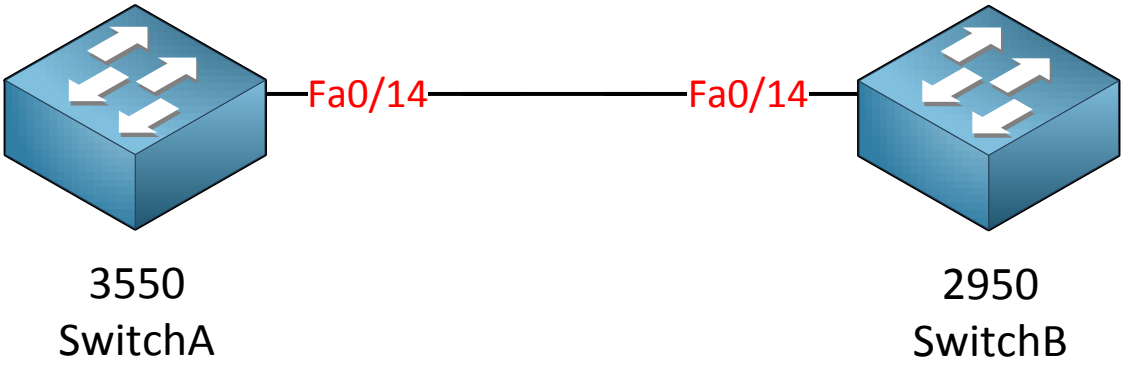
This is our trunk interface which is connected to SwitchA. You can see the operational mode is trunk mode.

```
SwitchB(config-if)#switchport mode ?
  access         Set trunking mode to ACCESS unconditionally
  dot1q-tunnel   set trunking mode to TUNNEL unconditionally
  dynamic        Set trunking mode to dynamically negotiate access or trunk
  private-vlan   Set private-vlan mode
  trunk          Set trunking mode to TRUNK unconditionally
```

If I go to the interface configuration to change the switchport mode you can see I have more options than access or trunk mode. There is also a **dynamic** method. Don't worry about the other options for now.

```
SwitchB(config-if)#switchport mode dynamic ?
  auto       Set trunking mode dynamic negotiation parameter to AUTO
  desirable  Set trunking mode dynamic negotiation parameter to DESIRABLE
```

We can choose between **dynamic auto** and **dynamic desirable.** Our switch will automatically find out if the interface should become an access or trunk port. So what's the difference between dynamic auto and dynamic desirable? Let's find out!



I'm going to play with the switchport mode on SwitchA and SwitchB and we'll see what the result will be.

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#switchport mode dynamic auto
```

```
SwitchA(config)#interface fa0/14
SwitchB(config-if)#switchport mode dynamic auto
```

First I'll change both interfaces to dynamic auto.

```
SwitchA(config-if)#do show interface f0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
```

```
SwitchB(config-if)#do show interface f0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
```

Our administrative mode is dynamic auto and as a result we now have an access port.

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#switchport mode dynamic desirable
```

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

```
SwitchA#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic desirable
Operational Mode: trunk
```

```
SwitchB#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic desirable
Operational Mode: trunk
```

Once we change both interfaces to dynamic desirable we end up with a trunk link. What do you think will happen if we mix the switchport types? Maybe dynamic auto on one side and dynamic desirable on the other side? Let's find out!

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#switchport mode dynamic desirable
```

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

```
SwitchA#show interfaces f0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic desirable
Operational Mode: trunk
```

```
SwitchB#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: trunk
```

It seems our switch has a strong desire to become a trunk. Let's see what happens with other combinations!

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#switchport mode dynamic auto
```

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

```
SwitchA#show interfaces f0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: trunk
```

```
SwitchB#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
```

Dynamic auto will prefer to become an access port but if the other interface has been configured as trunk we will end up with a trunk.

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#switchport mode dynamic auto
```

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

```
SwitchA#show interfaces f0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
```

```
SwitchB#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: static access
Operational Mode: static access
```

Configuring one side as dynamic auto and the other one as access and the result will be an access port.

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#switchport mode dynamic desirable
```

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

```
SwitchA#show interfaces f0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: dynamic desirable
Operational Mode: trunk
```

```
SwitchB#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
```

Dynamic desirable and trunk mode offers us a working trunk.

What do you think will happen if I set one interface in access mode and the other one as trunk? Doesn't sound like a good idea but let's push our luck:

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#switchport mode access
```

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

```
SwitchA#show interfaces f0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: static access
Operational Mode: trunk
```

```
SwitchB#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
```

```
SwitchA#
%SPANTREE-7-RECV_1Q_NON_TRUNK: Received 802.1Q BPDU on non trunk
FastEthernet0/14 VLAN1.
%SPANTREE-7-BLOCK_PORT_TYPE: Blocking FastEthernet0/14 on VLAN0001.
Inconsistent port type.
%SPANTREE-2-UNBLOCK_CONSIST_PORT: Unblocking FastEthernet0/14 on VLAN0001.
Port consistency restored.
```

As soon as I change the switchport mode I see these spanning-tree error messages on SwitchA. Spanning-tree receives an 802.1Q BPDU on an access port and doesn't like it. The interface goes into blocking mode for VLAN 1 and only 14 seconds later its unblocking VLAN 1 again. Does this mean we have connectivity even though this smells fishy?

```
SwitchA#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: static access
Operational Mode: static access
```

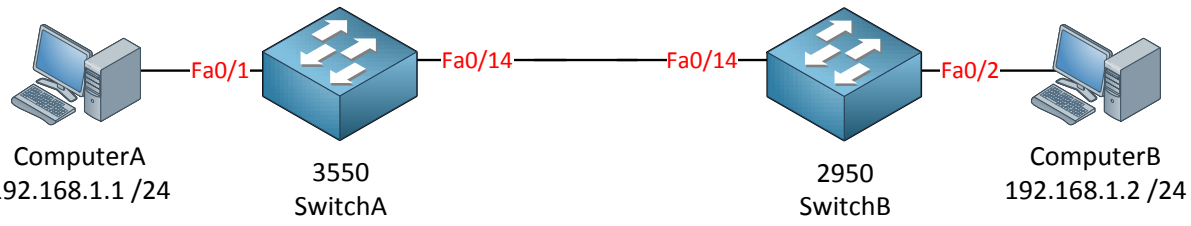
```
SwitchB#show interfaces fa0/14 switchport
Name: Fa0/14
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
```

This doesn't look good; let's continue by looking at the trunk…

```
SwitchA#show interfaces fa0/14 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/14      off              802.1q         not-trunking  1

Port        Vlans allowed on trunk
Fa0/14      1

Port        Vlans allowed and active in management domain
Fa0/14      1

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/14      1
```

```
SwitchB#show interfaces fa0/14 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/14      on               802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/14      1-50

Port        Vlans allowed and active in management domain
Fa0/14      1,50

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/14      50
```

Now this looks interesting. It seems SwitchA only allows VLAN 1 and SwitchB allows VLAN 1-50.



ComputerA and ComputerB are still in VLAN 50. Let's see if they can still reach each other:

```
C:\Documents and Settings\ComputerA>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),
```

No luck here…ComputerA and ComputerB are unable to reach each other. What if I move them to VLAN 1?

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

```
SwitchB(config)#interface fa0/2
SwitchB(config-if)#switchport access vlan 1
```

```
C:\Documents and Settings\ComputerA>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:
```

Excellent now it is working! So even though we have a mismatch between the switchport types we still have **limited connectivity** because **only VLAN 1 is allowed**.

Let me give you an overview of the different switchport modes and the result:

	Trunk	Access	Dynamic Auto	Dynamic Desirable
Trunk	Trunk	Limited	Trunk	Trunk
Access	Limited	Access	Access	Access
Dynamic Auto	Trunk	Access	Access	Trunk
Dynamic Desirable	Trunk	Access	Trunk	Trunk

Make sure you know the result of these combinations if you plan to do the CCNP SWITCH exam. I always like to think that the switch has a strong "desire" to become a trunk. Its wish will always be granted unless the other side has been configured as access port. The "A" in dynamic auto stands for "Access", it would like to become an access port but only if the other side also is configured as dynamic auto or access mode.

I recommend **never to use** the "dynamic" types. I want my interfaces to be in trunk OR access mode and I like to make the decision myself. Keep in mind that dynamic auto is the **default** on most modern switches which means it's possible to form a trunk with any interface on your switch automatically. Some of the older switches use dynamic desirable as the default. This is a security issue you should deal with! If I walk into your company building I could connect my laptop to any wall jack, boot up GNS3, form a trunk to your switch and I'll have access to all your VLANs…doesn't sound like a good idea right?

This is what I recommend for trunk interfaces:

```
Switch(config-if)#switchport mode trunk
Switch(config-if)#switchport nonegotiate
```

The negotiation of the switchport status by using dynamic auto or dynamic desirable is called **DTP (Dynamic Trunking Protocol).** You can disable it completely by using the **switchport nonegotiate** command.

One more thing about VLANs and trunks before we continue with VTP. I recommend changing the native VLAN to something else.

```
SwitchB#show interfaces fa0/14 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/14      on               802.1q         trunking      1
```

You can see that VLAN 1 is the default native VLAN on Cisco switches. Management protocols like CDP, DTP and LACP/PagP (Etherchannels…more on this later!)

use the native VLAN so for security reasons it might be a good idea to change it to something else:

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#switchport trunk native vlan 100
```

```
SwitchB(config)#interface fa0/14
SwitchB(config-if)#switchport trunk native vlan 100
```

This is how we change the native VLAN.

```
SwitchB#show interfaces fa0/14 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/14      on               802.1q         trunking      100
```

You can see the native VLAN is now VLAN 100.

By default the native VLAN is untagged when we use 802.1Q trunks. This can cause a security vulnerability (double tagging attack or "VLAN hopping") where we send double-tagged frames from one VLAN to another. One way of preventing this is by making sure that the native VLAN is tagged:

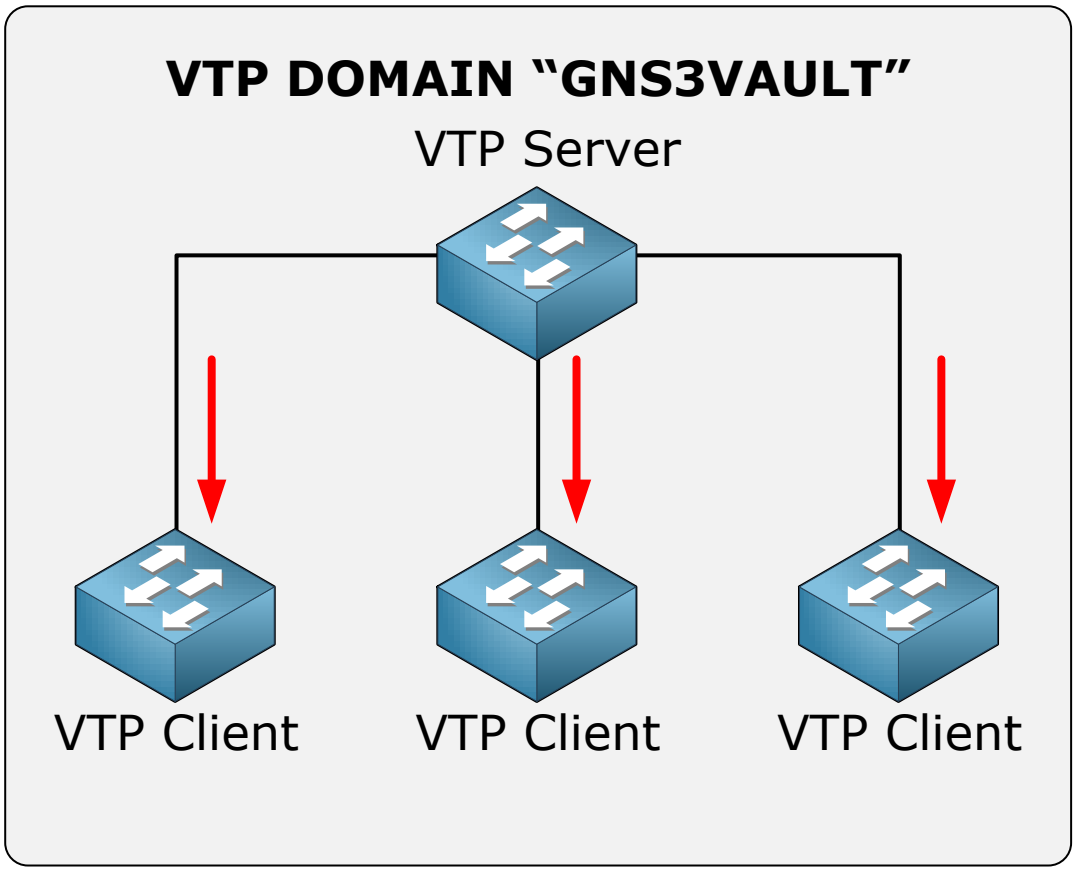
```
SwitchA(config)#vlan dot1q tag native
```

```
SwitchB(config)#vlan dot1q tag native
```

The **vlan dot1q tag native** command above will ensure that the native VLAN will be tagged on all trunks.

This is all that I have for you about VLANs and trunking. We still have to look at VTP (VLAN Trunking Protocol) which can help you to synchronize VLANS between switches.

Let's say you have a network with 20 switches and 50 VLANs. Normally you have to configure each switch separately and create those VLANs on each and every switch. That's a time consuming task so there is something to help us called VTP (Vlan Trunking Protocol). VTP will let you create VLANs on one switch and all the other switches will synchronize themselves.

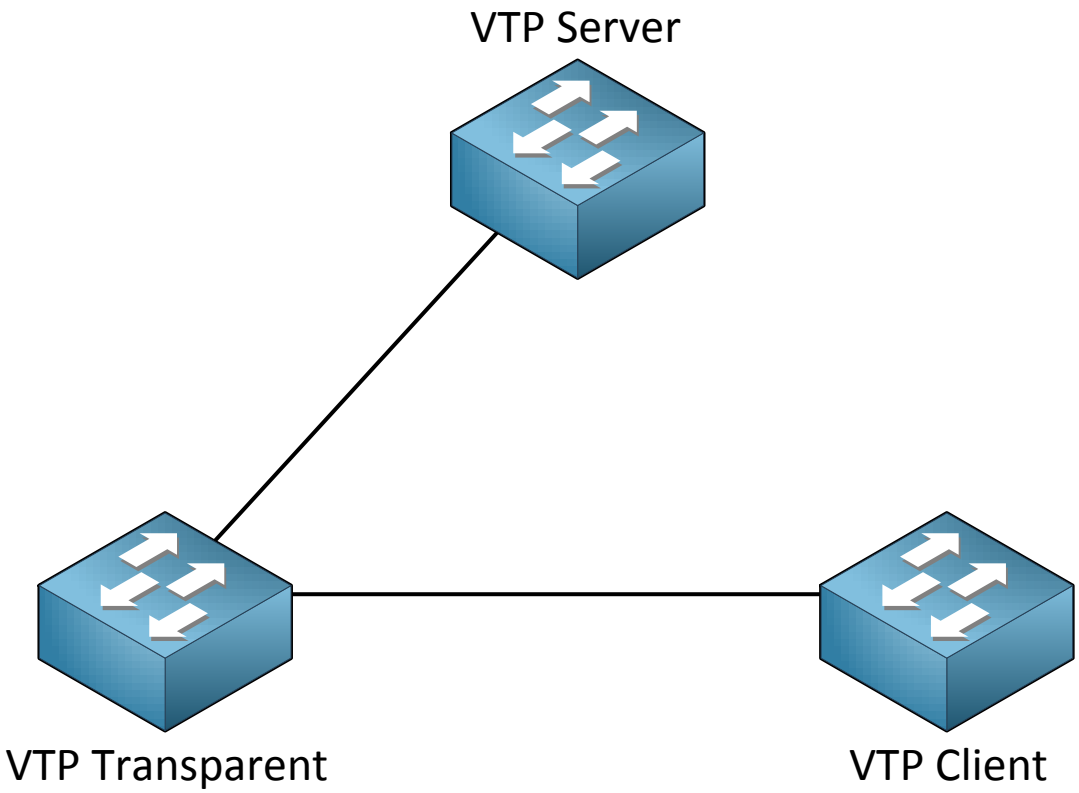


We have one VTP server which is the switch where you create / modify or delete vlans. The other switches are VTP clients. The VTP configuration has a revision number which will increase when you make a change. Every time you make a change on the VTP server this will be synchronized to the VTP clients. Oh and by the way you can have multiple VTP servers since it also functions as a VTP client so you can make changes on multiple switches in your network. In order to make VTP work you need to setup a VTP domain name which is something you can just make up, as long as you configure it to be the same on all your switches.

This is the short version of what I just described:

1. VTP adds / modifies / deletes vlans.
2. For every change the revision number will increase.
3. The latest advertisement will be sent to all VTP clients.
4. VTP clients will synchronize themselves with the latest information.

Besides the VTP server and VTP client there's also a VTP transparent which is a bit different, let me show you an example:



Our VTP Transparent will forward advertisements but will **not synchronize** itself. You can create vlans locally though which is impossible on the VTP client. Let's say you create vlan 20 on our VTP server, this is what will happen:

1. You create VLAN 20 on the VTP server.
2. The revision number will increase.
3. The VTP server will forward the latest advertisement which will reach the VTP transparent switch.
4. The VTP transparent will not synchronize itself but will forward the advertisement to the VTP client.
5. The VTP client will synchronize itself with the latest information

Here's an overview of the 3 VTP modes:

	VTP Server	VTP Client	VTP Transparent
Create/Modify/Delete Vlans	Yes	No	Only local
Synchronizes itself	Yes	Yes	No
Forwards advertisements	Yes	Yes	Yes

Should you use VTP? It might sound useful but VTP has a **huge security risk**…the problem with VTP is that a VTP server is also a VTP Client and any VTP client will synchronize itself with the highest revision number.

The following situation can happen with VTP:

You have a network with a single VTP server and a couple of VTP client switches, everything is working fine but one day you want to test some stuff and decide to take one of the VTP clients out of the network and put it in a lab environment.

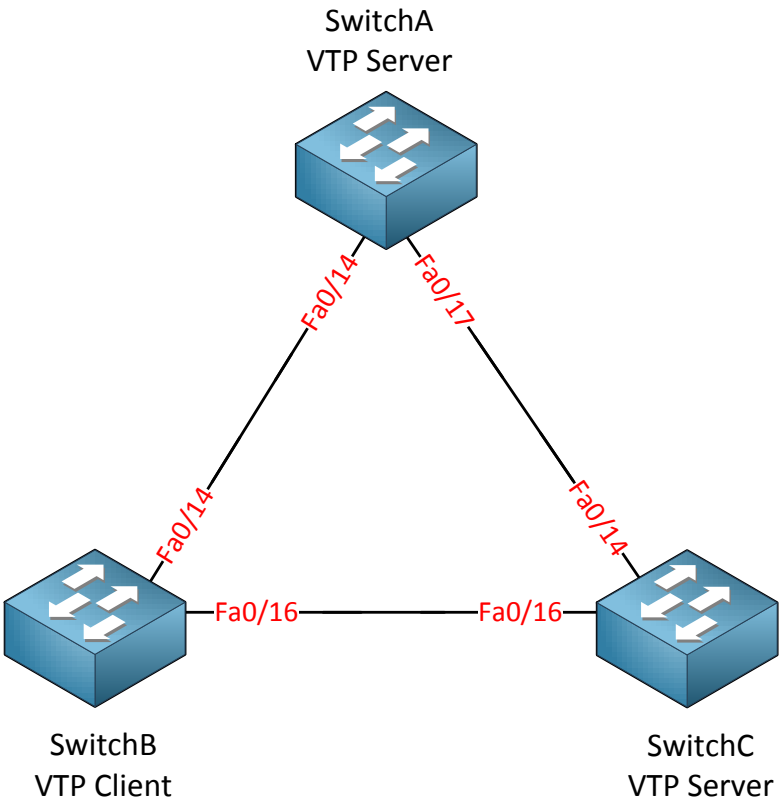
1. You take the VTP client switch out of the network.
2. You configure it so it's no longer a VTP Client but a VTP server.
3. You play around with VTP, create some vlans, and modify some.
4. Every time you make a change the revision number increases.
5. You are done playing…you delete all vlans.
6. You configure the switch from VTP Server to VTP Client.
7. You connect your switch to your production network.

What do you think the result will be? The revision number of VTP on the switch we played with is higher than the revision number on the switches of our production network. The VTP client will advertise its information to the other switches, they synchronize to the latest information and POOF all your vlans are gone! A VTP client can **overwrite** a VTP server if the revision number is higher because a VTP server is also a VTP client.

Yes I know this sounds silly but this is the way it works…very dangerous since you'll lose all your VLAN information. Your interfaces won't go back to VLAN 1 by default but will float around in no man's land…

VTP has two versions…1 and 2. The two versions are **incompatible** so make sure you use either version 1 or 2. Version 1 is the default. VTP version 2 offers a number of additional features:

- **Version dependent transparent mode**: When using VTP transparent mode, VTP version 1 matches the VTP version and domain name before it forwards VTP messages to other VTP switches. Version 2 forwards VTP messages without checking the version number.
- **Consistency checks**: VTP version 2 does consistency checks when you enter VTP or VLAN information. This is done to ensure no incorrect VLAN names or numbers are sent to other VTP switches. These checks don't apply on incoming VTP messages.
- **Token ring support**: You probably won't see it anymore but VTP version 2 supports token ring, VTP version 1 does not.
- **Unrecognized TLV support**: VTP version 2 will forward received VTP configuration change messages even if it doesn't understand some fields in the VTP message. VTP version 1 will drop VTP messages that it doesn't understand.



Let's take a look at the configuration of VTP. I will be using three switches for this task. I erased the VLAN database and the startup-configuration on all switches.

```
SwitchA#show vtp status
VTP Version                     : running VTP1 (VTP2 capable)
Configuration Revision          : 0
Maximum VLANs supported locally : 1005
Number of existing VLANs        : 5
VTP Operating Mode              : Server
VTP Domain Name                 :
VTP Pruning Mode                : Disabled
VTP V2 Mode                     : Disabled
VTP Traps Generation            : Disabled
MD5 digest                      : 0x57 0xCD 0x40 0x65 0x63 0x59 0x47 0xBD
Configuration last modified by 0.0.0.0 at 0-0-00 00:00:00
Local updater ID is 0.0.0.0 (no valid interface found)
```

```
SwitchB#show vtp status
VTP Version                     : running VTP1 (VTP2 capable)
Configuration Revision          : 0
Maximum VLANs supported locally : 1005
Number of existing VLANs        : 5
VTP Operating Mode              : Server
VTP Domain Name                 :
VTP Pruning Mode                : Disabled
VTP V2 Mode                     : Disabled
VTP Traps Generation            : Disabled
MD5 digest                      : 0x57 0xCD 0x40 0x65 0x63 0x59 0x47 0xBD
Configuration last modified by 0.0.0.0 at 0-0-00 00:00:00
Local updater ID is 0.0.0.0 (no valid interface found)
```

```
SwitchC#show vtp status
VTP Version                     : 2
Configuration Revision          : 0
Maximum VLANs supported locally : 1005
Number of existing VLANs        : 5
VTP Operating Mode              : Server
VTP Domain Name                 :
VTP Pruning Mode                : Disabled
VTP V2 Mode                     : Disabled
VTP Traps Generation            : Disabled
MD5 digest                      : 0x57 0xCD 0x40 0x65 0x63 0x59 0x47 0xBD
Configuration last modified by 0.0.0.0 at 0-0-00 00:00:00
Local updater ID is 0.0.0.0 (no valid interface found)
```

Depending on the switch model you will see a similar output if you use the **show vtp status** command. There's a couple of interesting things to see here:

- Configuration revision 0: Each time we add or remove VLANs this number will change. It's 0 at the moment since I haven't created or removed any VLANs.
- VTP Operating mode: the default is VTP server.
- VTP Pruning: this will help to prevent unnecessary traffic on your trunk links, more in this later.
- VTP V2 Mode: The switch is capable of running VTP version 2 but it's currently running VTP version 1.

```
SwitchA(config)#vlan 10
SwitchA(config-vlan)#name Printers
```

Let's create a VLAN on SwitchA and we'll see if anything changes…

```
SwitchA#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
1    default                           active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                 Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                 Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                 Fa0/13, Fa0/14, Fa0/15,
Fa0/22
                                                 Fa0/23, Fa0/24, Gi0/1, Gi0/2
10   Printers                          active
```

My new VLAN shows up in the VLAN database, so far so good…

```
SwitchA#show vtp status
VTP Version                     : running VTP1 (VTP2 capable)
Configuration Revision          : 1
```

You can see that the configuration revision has increased by one.

```
SwitchB#show vtp status
VTP Version                     : running VTP1 (VTP2 capable)
Configuration Revision          : 0
```

```
SwitchC#show vtp status
VTP Version                     : 2
Configuration Revision          : 0
```

Unfortunately nothing has changed on SwitchB and SwitchC. This is because we need to configure a **VTP domain-name** before it starts working.

```
SwitchB#debug sw-vlan vtp events
vtp events debugging is on
```

```
SwitchC#debug sw-vlan vtp events
vtp events debugging is on
```

Before I change the domain-name I'm going to enable a debug using the **debug sw-vlan vtp events** command. This way we can see in real-time what is going on.

```
SwitchA(config)#vtp domain GNS3VAULT
Changing VTP domain name from NULL to GNS3VAULT
```

```
SwitchB#
VTP LOG RUNTIME: Summary packet received in NULL domain state
VTP LOG RUNTIME: Summary packet received, domain = GNS3VAULT, rev = 1,
followers = 1, length 77, trunk Fa0/16
VTP LOG RUNTIME: Transitioning from NULL to GNS3VAULT domain
VTP LOG RUNTIME: Summary packet rev 1 greater than domain GNS3VAULT rev 0
```

You will see the following debug information on SwitchB and SwitchC; there are two interesting things we can see here:

- The switch receives a VTP packet from domain "GNS3VAULT" and decides to change its own domain-name from "NULL" (nothing) to "GNS3VAULT". It will only change the domain-name if it doesn't have a domain-name.
- The switch sees that the VTP packet has a higher revision number (1) than what it currently has (0) and as a result it will synchronize itself.

```
SwitchB#no debug all
All possible debugging has been turned off
```

```
SwitchC#no debug all
All possible debugging has been turned off
```

Make sure to disable the debug output before you get flooded with information.

```
SwitchB#show vtp status
VTP Version                     : running VTP1 (VTP2 capable)
Configuration Revision          : 1
```

```
SwitchC#show vtp status
VTP Version                     : 2
Configuration Revision          : 1
```

The revision number on SwitchB and SwitchC is now "1".

```
SwitchB#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/13, Fa0/14, Fa0/15,
                                                Fa0/23, Fa0/24, Gi0/1, Gi0/2
10   Printers                         active
```

```
SwitchC#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/20, Fa0/22, Fa0/23,
                                                Gi0/1, Gi0/2
10   Printers                         active
```

The show vlan command tells us that SwitchB and SwitchC have learned VLAN 10 through VTP.
Since all switches are in VTP Server mode I can create VLANs on any switch and they should all synchronize:

```
SwitchB(config)#vlan 20
SwitchB(config-vlan)#name Servers
```

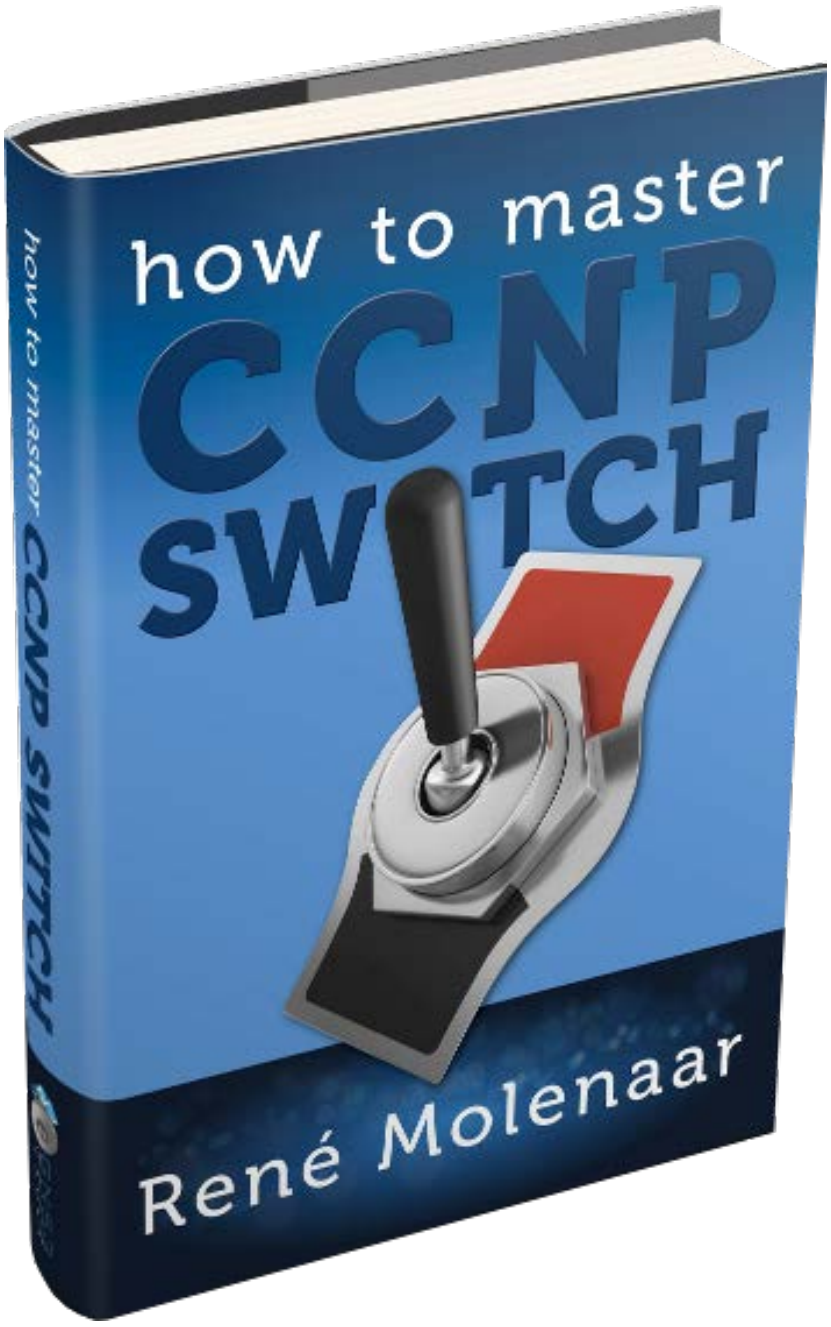
```
SwitchC(config)#vlan 30
SwitchC(config-vlan)#name Management
```

Let's create VLAN 20 on SwitchB and VLAN 30 on SwitchC.

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

Click on the link below to get the full version.

Get How to Master CCNP SWITCH Today



```
SwitchA#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
10   Printers                          active
20   Servers                           active
30   Management                        active
```

```
SwitchB#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
10   Printers                          active
20   Servers                           active
30   Management                        active
```

```
SwitchC#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
10   Printers                          active
20   Servers                           active
30   Management                        active
```

As you can see all switches know about the VLANs. What about the revision number? Did it change?

```
SwitchA#show vtp status
VTP Version                     : running VTP1 (VTP2 capable)
Configuration Revision          : 3
```

```
SwitchB#show vtp status
VTP Version                     : running VTP1 (VTP2 capable)
Configuration Revision          : 3
```

```
SwitchC#show vtp status
VTP Version                     : 2
Configuration Revision          : 3
```

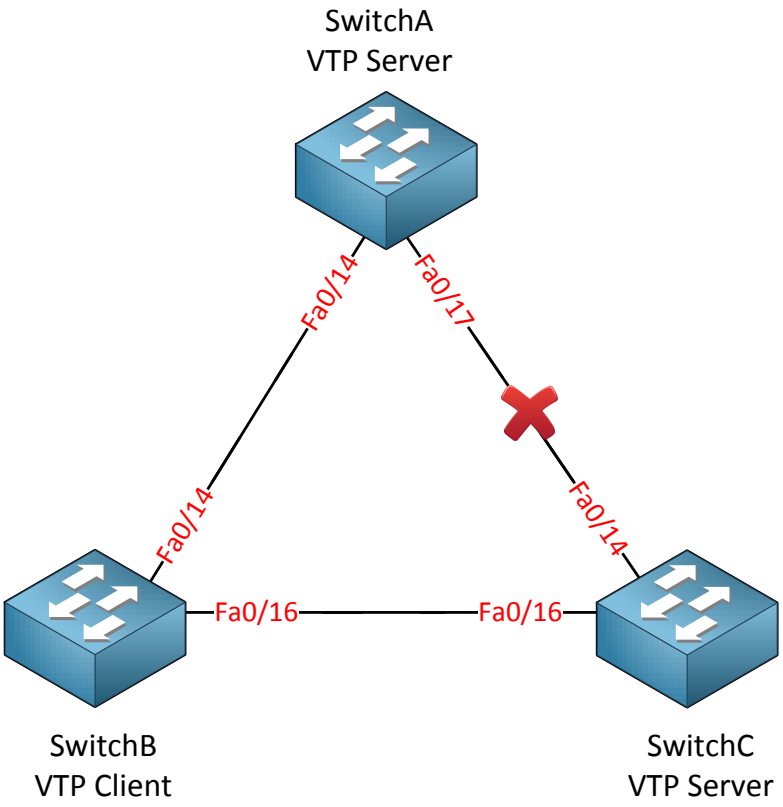
Each time I create another VLAN the revision number increases by one.

Let's change the VTP mode on SwitchB to see what it does.

```
SwitchB(config)#vtp mode client
Setting device to VTP CLIENT mode.
```

```
SwitchB#show vtp status
VTP Version                     : running VTP1 (VTP2 capable)
Configuration Revision          : 3
Maximum VLANs supported locally : 1005
Number of existing VLANs        : 7
VTP Operating Mode              : Client
```

It's now running in VTP Client mode.



Right now SwitchA and SwitchC are in VTP Server mode. SwitchB is running VTP Client mode. I have disconnected the link between SwitchA and SwitchC so there is no direct connection between them.

```
SwitchA(config)#vlan 40
SwitchA(config-vlan)#name Engineering
```

I'll create another VLAN on SwitchA so we can see if SwitchB and SwitchC will learn it.

```
SwitchB#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--
10   Printers                         active
20   Servers                          active
30   Management                       active
40   Engineering                      active
```

SwitchB learns about VLAN 40 through SwitchA.

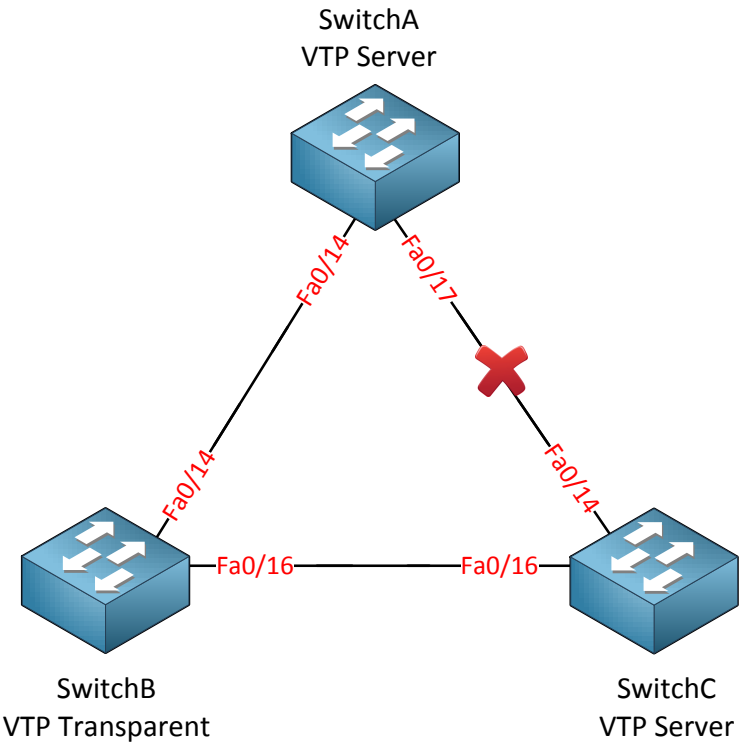
```
SwitchC#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--
10   Printers                         active
20   Servers                          active
30   Management                       active
40   Engineering                      active
```

SwitchC learns about VLAN 40 through SwitchB. SwitchB as a VTP client will synchronize itself but it will also forward VTP advertisements.

```
SwitchB(config)#vlan 50
%VTP VLAN configuration not allowed when device is in CLIENT mode.
```

A switch running in VTP Client mode is unable to create VLANs so that's why I get this error if I try to create one.

What about the VTP Transparent mode? That's the last one we have to try…



I'll change SwitchB to VTP Transparent mode and the link between SwitchA and SwitchC is still disconnected.

```
SwitchB(config)#vtp mode transparent
Setting device to VTP TRANSPARENT mode.
```

This is how we change SwitchB to VTP Transparent mode.

```
SwitchA(config)#vlan 50
SwitchA(config-vlan)#name Research
```

Let's create VLAN 50 for this experiment on SwitchA.

```
SwitchA#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--
10   Printers                         active
20   Servers                          active
30   Management                       active
40   Engineering                      active
50   Research                         active
```

It shows up on SwitchA as expected.

```
SwitchB#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--
10   Printers                         active
20   Servers                          active
30   Management                       active
40   Engineering                      active
```

It doesn't show up on SwitchB because it's in VTP transparent mode and doesn't synchronize itself.

```
SwitchC#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--
10   Printers                         active
20   Servers                          active
30   Management                       active
40   Engineering                      active
50   Research                         active
```

It does show up on SwitchC! A switch in VTP Transparent mode will **not synchronize itself** but it will **forward VTP advertisements** to other switches so they can synchronize themselves.

What will happen if I create a VLAN on SwitchB? Let's find out!

```
SwitchB(config)#vlan 60
SwitchB(config-vlan)#name Cameras
```

```
SwitchB#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--
10   Printers                         active
20   Servers                          active
30   Management                       active
40   Engineering                      active
60   Cameras                          active
```

We can create this new VLAN on SwitchB without any trouble. It's in VTP Transparent mode so we can do this.

```
SwitchA#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--
10   Printers                         active
20   Servers                          active
30   Management                       active
40   Engineering                      active
50   Research                         active
```

```
SwitchC#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--
10   Printers                         active
20   Servers                          active
30   Management                       active
40   Engineering                      active
50   Research                         active
```

VLAN 60 doesn't show up on SwitchA and SwitchC because SwitchB is in VTP Transparent mode. SwitchB will not advertise its VLANs because they are only **known locally.**

Is there anything else you need to know about VTP Transparent mode?

```
SwitchB#show running-config
Building configuration...

vlan 10
 name Printers
!
vlan 20
 name Servers
!
vlan 30
 name Management
!
vlan 40
 name Engineering
!
vlan 60
 name Cameras
```

There's a difference between VTP Transparent mode VS Server/Client mode. If you look at the running-config you will see that VTP Transparent stores all VLAN information in the running-config. VTP Server and Client mode store their information in the VLAN database (vlan.dat on your flash memory).

I haven't showed you yet how to change the VTP version, for the configurations you just witnessed it doesn't matter what version you use. Here's how to do it:

```
SwitchA(config)#vtp version 2
```

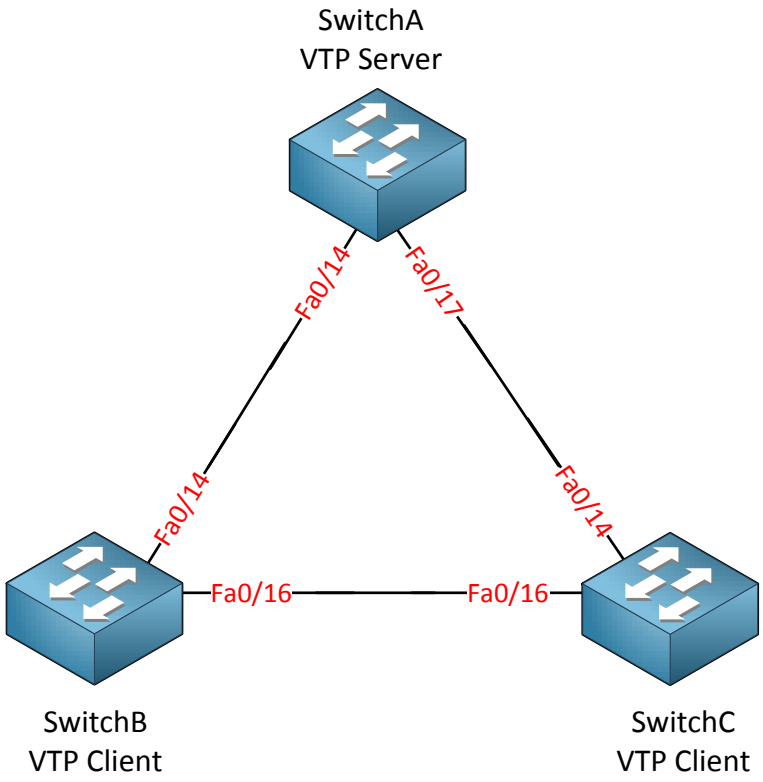
```
SwitchB(config)#vtp version 2
```

```
SwitchC(config)#vtp version 2
```

Use the **vtp version** command to change the VTP version. Normally you only have to configure this on the VTP server, the VTP clients will sync themselves and change their version. You can verify our configuration like this:

```
SwitchA#show vtp status | include Version
VTP Version                     : running VTP2
```

If you understand the difference between VTP Server, Client and Transparent mode…good! There's one more thing I want to show to you about VTP.



I'm going to demonstrate the danger of VTP as I explained before. A VTP client can overwrite a VTP server if the revision number of the VTP client is higher. I'm using the same topology but this time SwitchA is the VTP Server and SwitchB and SwitchC are VTP Clients.

```
SwitchA(config)#vtp mode server
Device mode already VTP SERVER.
SwitchA(config)#vtp domain GNS3VAULT
Changing VTP domain name from NULL to GNS3VAULT
```

```
SwitchB(config)#vtp mode client
Setting device to VTP CLIENT mode.
```

```
SwitchC(config)#vtp mode client
Setting device to VTP CLIENT mode.
```

First I change the domain-name and configure the correct VTP modes.

```
SwitchA(config)#vlan 10
SwitchA(config-vlan)#name Printers
SwitchA(config)#vlan 20
SwitchA(config-vlan)#name Servers
SwitchA(config)#vlan 30
SwitchA(config-vlan)#name Management
```

Next step is to create a couple of VLANS.

```
SwitchA(config)#interface fa0/1
SwitchA(config-if)#switchport mode access
SwitchA(config-if)#switchport access vlan 10
```

I will configure one (random) interface so it's in VLAN 10.

```
SwitchA#show vtp status
VTP Version                     : running VTP2
Configuration Revision          : 4
```

```
SwitchB#show vtp status
VTP Version                     : running VTP2
Configuration Revision          : 4
```

```
SwitchC#show vtp status
VTP Version                     : 2
Configuration Revision          : 4
```

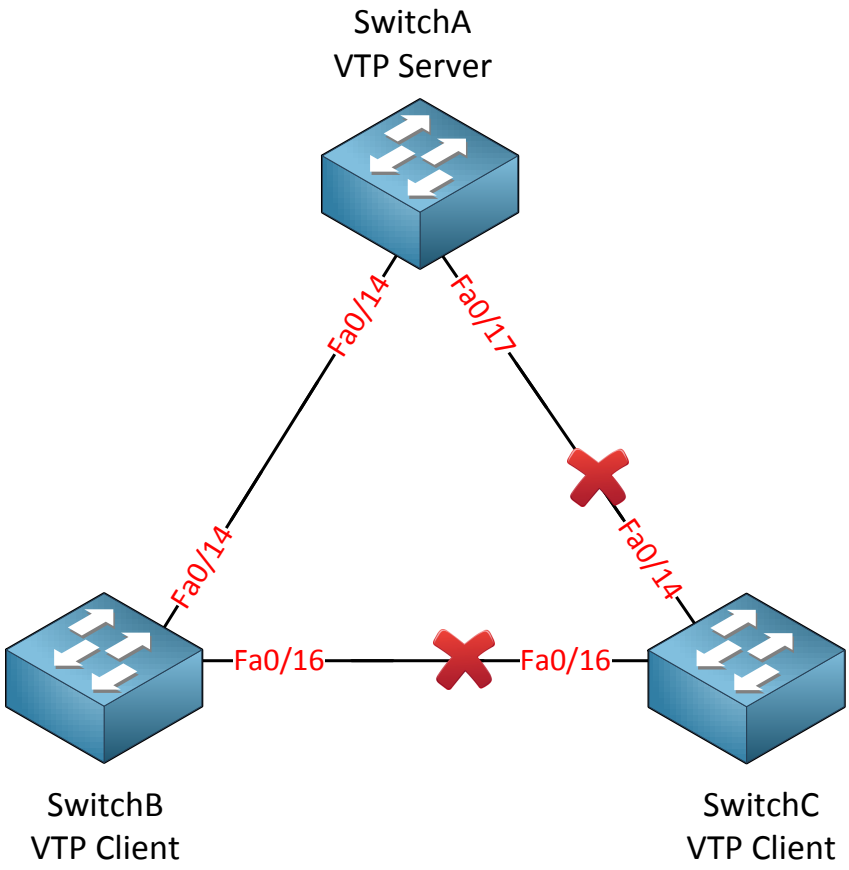
All switches currently have the same revision number.

```
SwitchA#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
1    default                           active    Fa0/2, Fa0/3, Fa0/4, Fa0/5
10   Printers                          active    Fa0/1
20   Servers                           active
30   Management                        active
```

```
SwitchB#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
1    default                           active    Fa0/2, Fa0/3, Fa0/4, Fa0/5
10   Printers                          active
20   Servers                           active
30   Management                        active
```

```
SwitchC#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
1    default                           active    Fa0/2, Fa0/3, Fa0/4, Fa0/5
10   Printers                          active
20   Servers                           active
30   Management                        active
```

All switches are up-to-date with the latest VLAN information. Note that Fa0/1 on SwitchA is in VLAN 10 at this moment.



Now I'm going to shut down the interfaces on SwitchC connecting SwitchA and SwitchB. This could happen if you want to remove a switch from your production network and temporarily use it in a lab.

```
SwitchC(config)#interface fa0/14
SwitchC(config-if)#shutdown
SwitchC(config)#interface fa0/16
SwitchC(config-if)#shutdown
```

I will change SwitchC from VTP Client mode to VTP Server mode:

```
SwitchC(config)#vtp mode server
Setting device to VTP SERVER mode
```

Easy enough! Let's add some VLANs:

```
SwitchC(config)#vlan 70
SwitchC(config-vlan)#name Lab
SwitchC(config)#vlan 80
SwitchC(config-vlan)#name Experiment
```

```
SwitchC#show vtp status
VTP Version                     : 2
Configuration Revision          : 6
```

After adding the VLANs you can see that the VTP revision number has increased.

```
SwitchC(config)#no vlan 10
SwitchC(config)#no vlan 20
SwitchC(config)#no vlan 30
SwitchC(config)#no vlan 70
SwitchC(config)#no vlan 80
```

```
SwitchC(config)#vtp mode client
Setting device to VTP CLIENT mode.
```

After playing with my lab I'm going to erase the VLANs and change the switch back to VTP Client mode so we can return it to the production network.

```
SwitchC#show vtp status
VTP Version                     : 2
Configuration Revision          : 11
```

Note that after deleting the VLANs the VTP revision number increased even more.

```
SwitchC(config)#interface fa0/14
SwitchC(config-if)#no shutdown
SwitchC(config)#interface fa0/16
SwitchC(config-if)#no shutdown
```

Let's do a "no shutdown" on the interfaces and return SwitchC to the production network.

```
SwitchA#show vtp status
VTP Version                     : running VTP2
Configuration Revision          : 11
```

```
SwitchB#show vtp status
VTP Version                     : running VTP2
Configuration Revision          : 11
```

Ugh…this doesn't look good. SwitchA and SwitchB now have the same revision number as SwitchC. This is the moment where you start to get nervous before you type in the next command…

```
SwitchA#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
1    default                           active    Fa0/2, Fa0/3, Fa0/4, Fa0/5
```

```
SwitchB#show vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
--
1    default                           active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
```

OUCH! All VLAN information is lost since SwitchA and SwitchB are synchronized to the latest information from SwitchC. This is the moment where your relaxing Monday morning turns into a horrible day…if you are lucky the support ticket system doesn't work anymore…if you are using VoIP than there's a chance your phones don't work anymore and you just have to wait till a mob of angry users will ram your door because they blame you for not being able to reach Facebook anymore…

Ok maybe I'm exaggerating a bit but you get the idea. If you have a big flat network with lots of switches and VLANs than this would be a disaster. I would advise to use VTP Transparent mode on all your switches and create VLANs locally!

If you do want to use VTP Server / Client mode you need to make sure you **reset the revision number**:

- Changing the **domain-name** will reset the revision number.
- Deleting the vlan.dat file on your flash memory will reset the revision number.

What happens to interfaces when you delete a VLAN? Take a close look at the show vlan command on SwitchA on the previous page. When you delete a VLAN all interfaces are in 'no-man's land'. They don't return to VLAN 1…you'll have to re-assign them yourself!

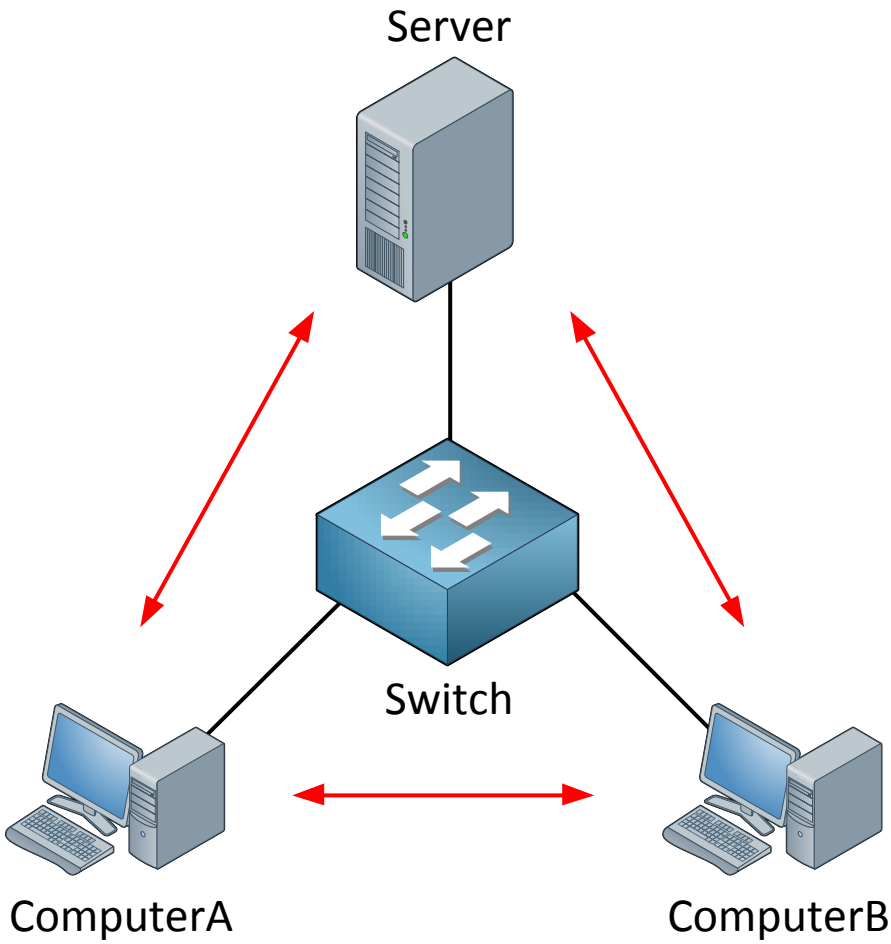
This is the end of the VLAN, Trunks and VTP chapter. If you want to get some practice I recommend you to take a look at the following labs:

http://gns3vault.com/Switching/vlans-and-trunks.html

http://gns3vault.com/Switching/vtp-vlan-trunking-protocol.html

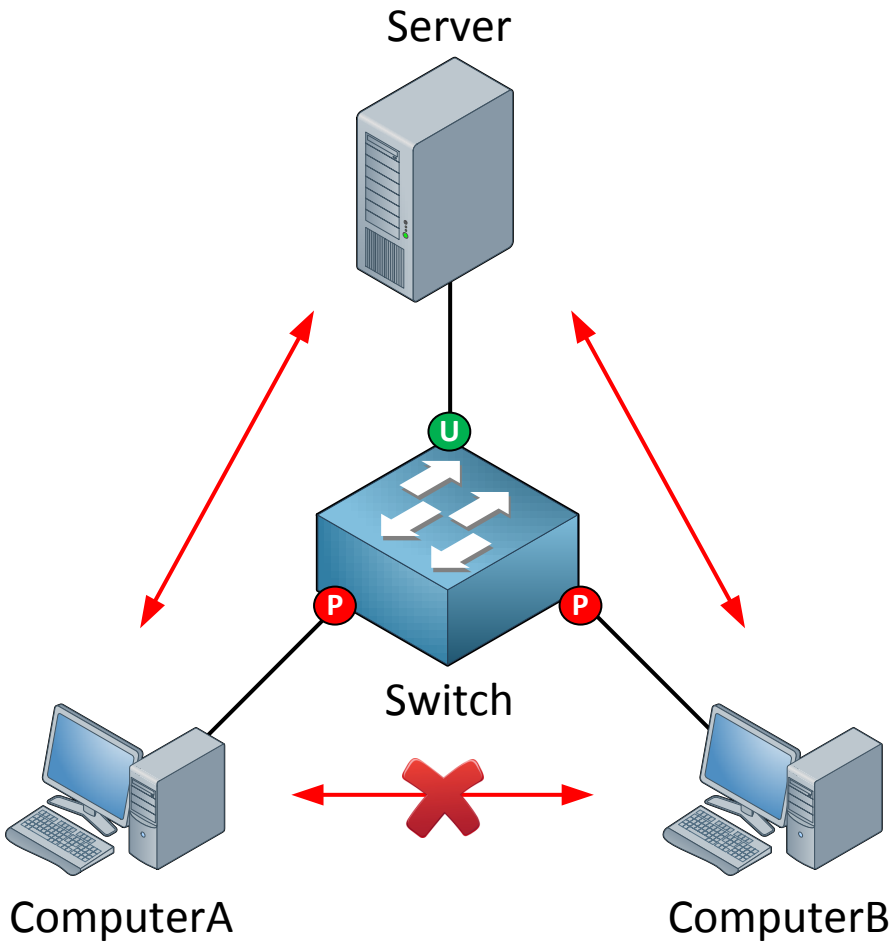
3. Private VLANs

If you studied CCNA then the previous chapter about VLANs, trunks and VTP was probably very familiar to you. In this chapter we will take a look at the **protected port** and **private VLANs.**

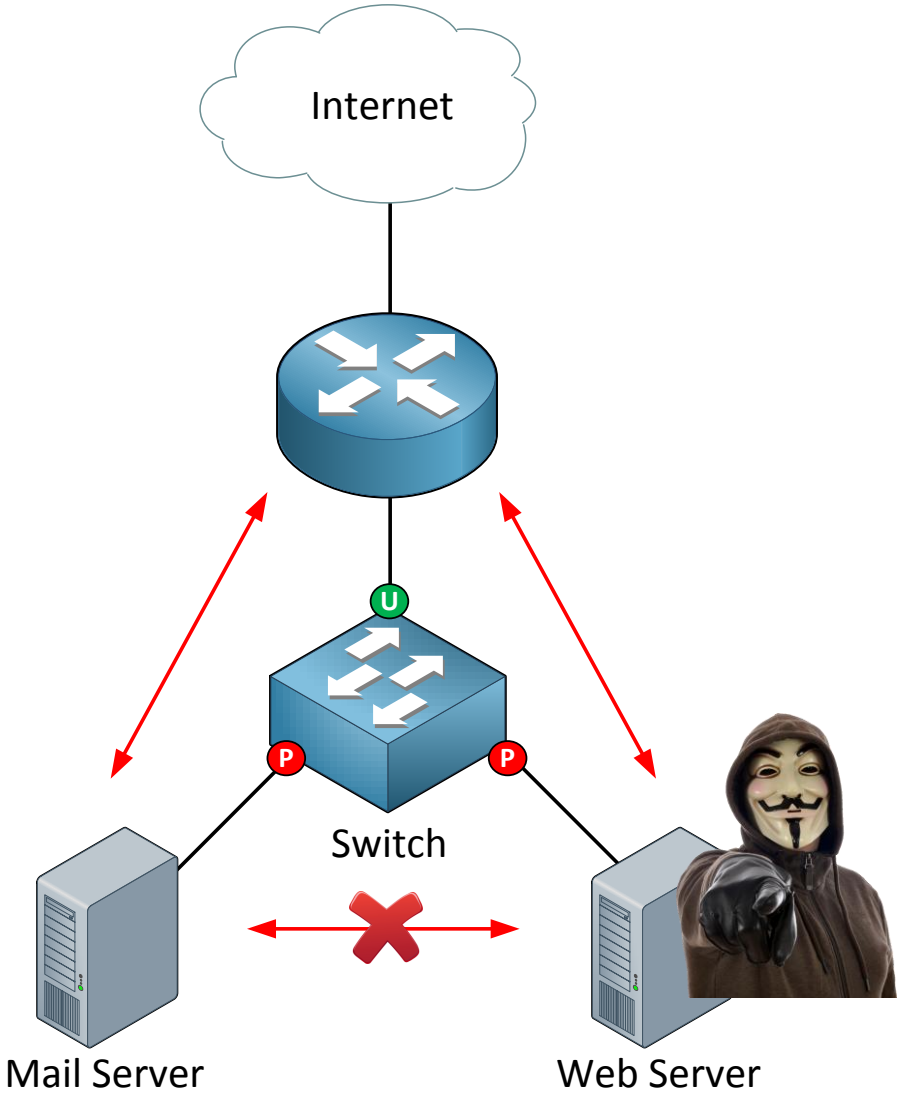


Take a look at the picture above. We have two computers, one switch and one server. Nothing fancy here…everything is in one VLAN and the two computers and server can communicate with each other.

What if I want to enhance security and ensure that ComputerA and ComputerB can only reach the server but not each other? This makes perfect sense in a client-server network. Normally there is no need for computers to connect to each other (unless Bob and Jane are secretly using shared folders on their computers without permission from the windows administrator).

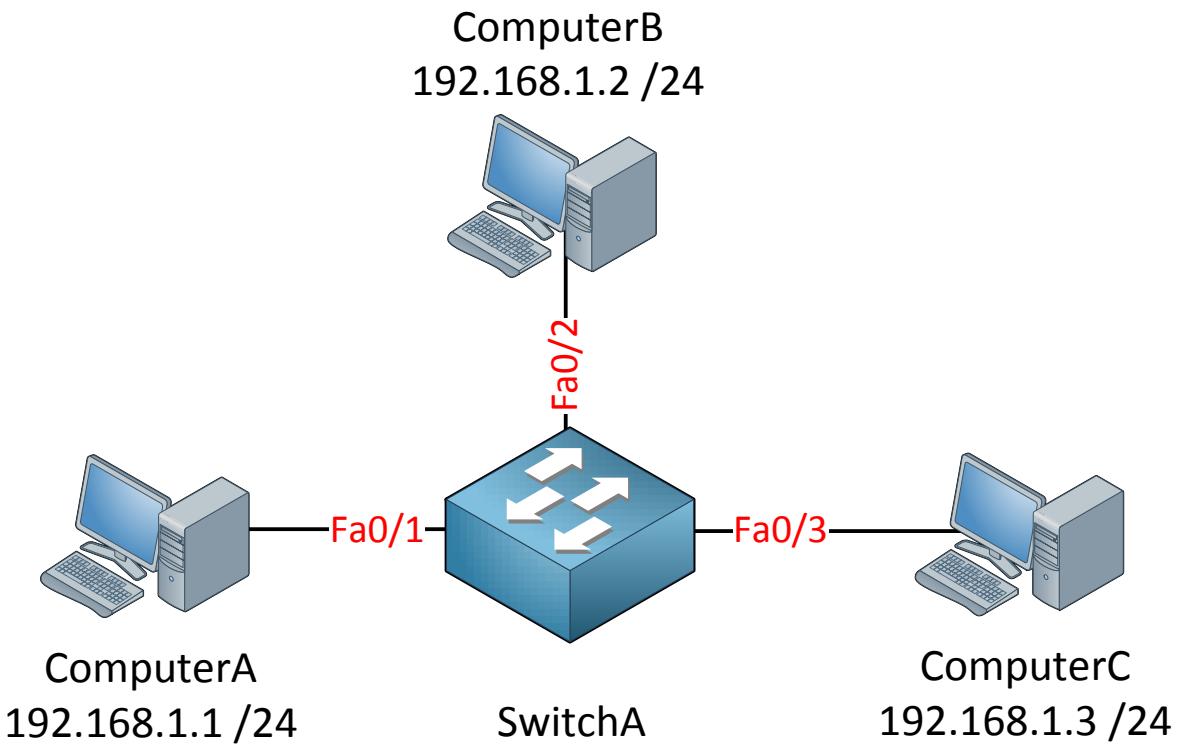


We can ensure ComputerA and ComputerB are unable to communicate with each other by using **protected ports**. By default all switch ports are unprotected.



You might also want to use it for servers in your network. If a ~~hacker~~ freedom fighter takes over your web server you can reduce the attack surface by preventing them from connecting to other servers in your network.

Let's take a switch and configure some protected ports!



This is the topology:

- Three computers in one subnet (192.168.1.0 /24)
- All computers are in the same VLAN (VLAN 1 by default).
- Default configuration on the switch.

```
C:\Documents and Settings\ComputerA>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
```

```
C:\Documents and Settings\ComputerA>ping 192.168.1.3

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.3: bytes=32 time<1ms TTL=128
```

```
C:\Documents and Settings\ComputerC>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
```

By sending a couple of pings between the computers we can verify that we have full reachability at this moment.

```
SwitchA(config)#interface fa0/1
SwitchA(config-if)#switchport protected
```

```
SwitchA(config)#interface fa0/3
SwitchA(config-if)#switchport protected
```

The interfaces connected to ComputerA and ComputerC are now protected. Interface fa0/2 to ComputerB is still unprotected.

```
SwitchB#show interfaces fa0/1 switchport
Name: Fa0/1
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: down
Administrative Trunking Encapsulation: negotiate
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk associations: none
Administrative private-vlan trunk mappings: none
Operational private-vlan: none
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL
Protected: true
```

We can verify our configuration by using the show interfaces switchport command. Close to the bottom of the output you will find:

Protected: true

```
SwitchB#show interfaces fa0/1 switchport | include Protected
Protected: true
```

If you know what you are looking for in the output it's easier to use the "include" command. This saves you hammering on the enter or space button of your keyboard…

```
C:\Documents and Settings\ComputerA>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
```

```
C:\Documents and Settings\ComputerC>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
```

ComputerA and ComputerC are still able to reach ComputerB.

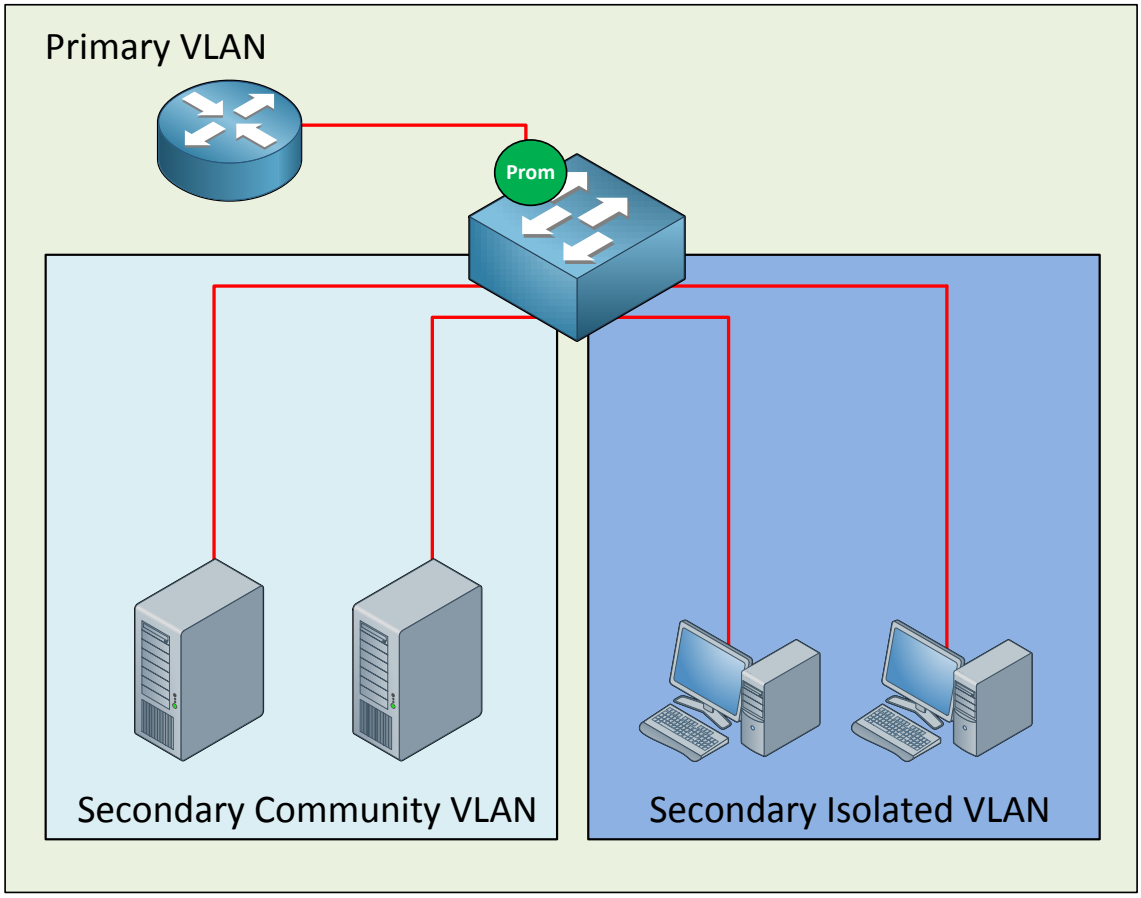
```
C:\Documents and Settings\ComputerA>ping 192.168.1.3

Pinging 192.168.1.2 with 32 bytes of data:

Request timed out.

Ping statistics for 192.168.1.2:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),
```

```
C:\Documents and Settings\ComputerC>ping 192.168.1.1

Pinging 192.168.1.2 with 32 bytes of data:

Request timed out.

Ping statistics for 192.168.1.2:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),
```

ComputerA and ComputerC are unable to reach each other now.

Protected port ←→ Unprotected = **working**
Protected port ←→ Protected port = **not working**

The protected port is pretty neat and as you have seen very easy to configure however it is very limited. In the last part of this chapter we'll take a look at **private VLANs.** Private VLANs are like protected ports on steroids!



Many network students believe private VLANs are very complex when they see this for the first time. I'm going to break it down and explain to you how it works.

The private VLAN always has one **primary VLAN.** Within the primary VLAN you will find the **promiscuous port**. In my picture above you can see that there's a router connected to a promiscuous port. **All other ports are able to communicate** with the promiscuous port.

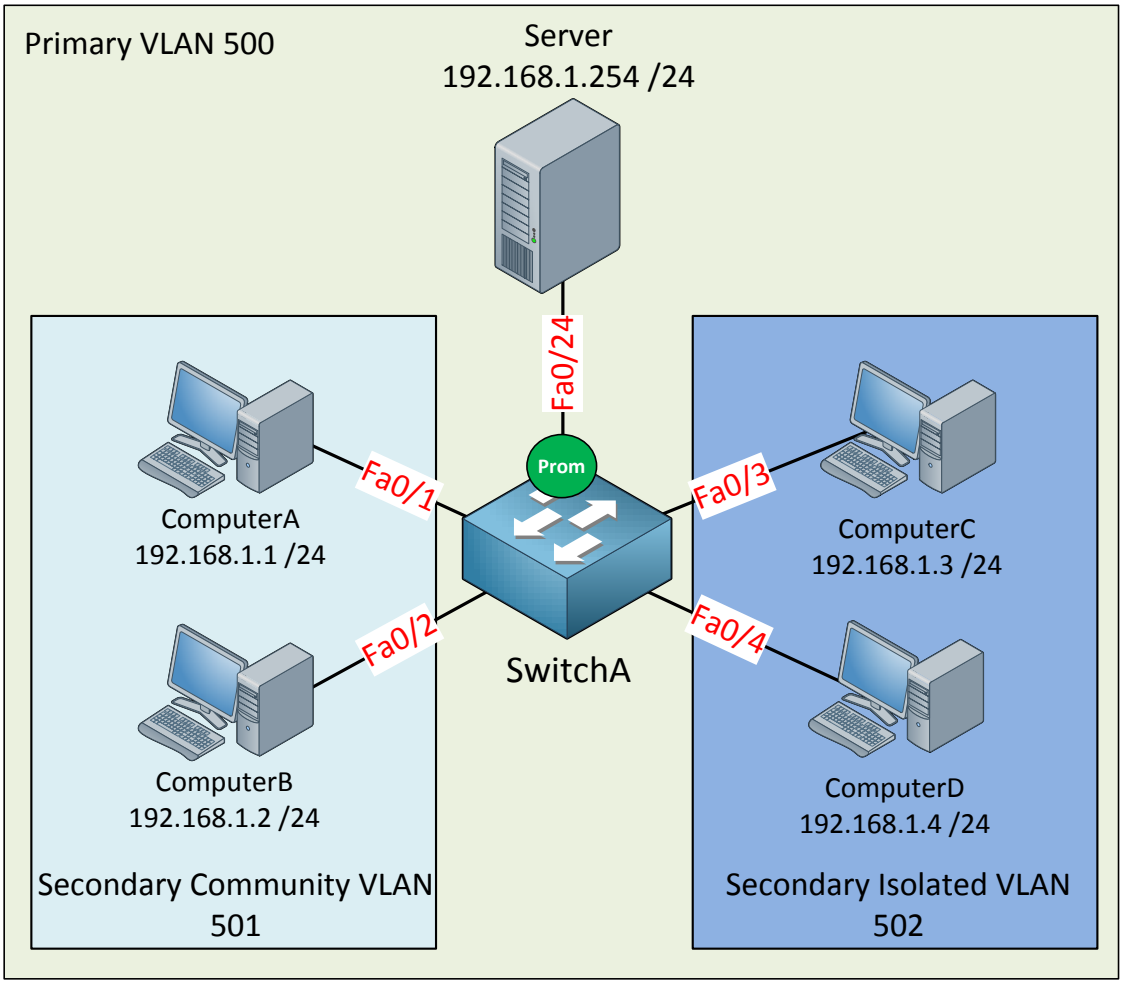
Within the primary VLAN you will encounter one or more **secondary VLANs**, there are two types:

- **Community VLAN:** All ports within the community VLAN are **able** to communicate with each other and the promiscuous port.
- **Isolated VLAN:** All ports within the isolated VLAN are **unable** to communicate with each other but they can communicate with the promiscuous port.

The names for these secondary VLANs are well chosen if you ask me. In a *community* everyone is able to talk to each other. When you are *isolated* you can only talk to yourself or in case of our private VLANs…the *promiscuous port.*

Secondary VLANS can always communicate with the promiscuous port but they can never communicate with **other secondary VLANs!**

Are you following me so far? If so…good! If you are still a little fuzzy, don't worry. I'm going to show you the configuration and demonstrate you how this works.



Let me sum up what we have here:

- The primary VLAN has number 500.
- The secondary community VLAN has number 501.
- The secondary isolated VLAN has number 502.
- I just made up these VLAN numbers; you can use whatever you like.
- ComputerA and ComputerB in the community VLAN should be able to reach each other and also the server connected to the promiscuous port.
- ComputerC and ComputerD in the isolated VLAN can only communicate with the server on the promiscuous port.
- The server should be able to reach all ports.

```
SwitchA(config)#vtp mode transparent
Setting device to VTP TRANSPARENT mode.
```

Configuring private VLANs requires us to **change the VTP mode to Transparent**.

```
SwitchA(config)#vlan 501
SwitchA(config-vlan)#private-vlan community
SwitchA(config-vlan)#vlan 500
SwitchA(config-vlan)#private-vlan primary
SwitchA(config-vlan)#private-vlan association add 501
```

Let's start with the configuration of the community VLAN. First I create VLAN 501 and tell the switch that this is a community VLAN by typing the **private-vlan community** command. Secondly I am creating VLAN 500 and configuring it as the primary VLAN with the **private-vlan primary** command. Last but not least I need to tell the switch that VLAN 501 is a secondary VLAN by using the **private-vlan association** command.

```
SwitchA(config)#interface range fa0/1 - 2
SwitchA(config-if-range)#switchport mode private-vlan host
SwitchA(config-if-range)#switchport private-vlan host-association 500 501
```

Interface fa0/1 and fa0/2 are connected to ComputerA and ComputerB and belong to the community VLAN 501. On the interface level I need to tell the switch that these are host ports by issuing the **switchport mode private-vlan host** command. I also have to use the **switchport private-vlan host-association** command to tell the switch that VLAN 500 is the primary VLAN and 501 is the secondary VLAN.

```
SwitchA(config)#interface fa0/24
SwitchA(config-if)#switchport mode private-vlan promiscuous
SwitchA(config-if)#switchport private-vlan mapping 500 501
```

This is how I configure the promiscuous port. First I have to tell the switch that fa0/24 is a promiscuous port by typing the **switchport mode private-vlan promiscuous** command. I also have to map the VLANs by using the **switchport private-vlan mapping** command.

```
SwitchA#show interfaces fastEthernet 0/1 switchport
Name: Fa0/1
Switchport: Enabled
Administrative Mode: private-vlan host
Operational Mode: down
Administrative Trunking Encapsulation: negotiate
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: 500 (VLAN0500) 501 (VLAN0501)
Administrative private-vlan mapping: none
```

We can verify our configuration by looking at the switchport information.

Here is the output for fa0/1.

```
SwitchA#show interfaces fastEthernet 0/2 switchport | include host-as
Administrative private-vlan host-association: 500 (VLAN0500) 501 (VLAN0501)
```

Interface fa0/2 has the same configuration as fa0/1.

```
SwitchA#show interface fa0/24 switchport
Name: Fa0/24
Switchport: Enabled
Administrative Mode: private-vlan promiscuous
Operational Mode: private-vlan promiscuous
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: 500 (VLAN0500) 501 (VLAN0501)
```

Here is the switchport information for fa0/24 (our promiscuous port). You can see the mapping information.

```
SwitchA#show vlan private-vlan

Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------------
-
500     501       community         Fa0/1, Fa0/2, Fa0/24
```

The **show vlan private-vlan** command gives us valuable information. You can see that VLAN 500 is the primary VLAN and 501 is the secondary VLAN. It also tells us whether the VLAN is a community or isolated VLAN the ports.

```
SwitchA#show vlan private-vlan type

Vlan Type
---- -----------------
500  primary
501  community
```

I also like the **show vlan private-vlan type** command because it gives us a quick overview of the private VLANs.

So what's the result of this configuration?

If everything is OK we should now have a working community VLAN…let's find out!

```
C:\Documents and Settings\ComputerA>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
```

ComputerA is able to reach ComputerB.

```
C:\Documents and Settings\ComputerA>ping 192.168.1.254

Pinging 192.168.1.254 with 32 bytes of data:

Reply from 192.168.1.254: bytes=32 time<1ms TTL=128
```

ComputerA can also reach the server behind the promiscuous port.

```
C:\Documents and Settings\Server>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
```

The server is able to reach ComputerB. Great! Our community VLAN seems to be up and running. Let's continue with the configuration of the isolated VLAN.

```
SwitchA(config)#vlan 502
SwitchA(config-vlan)#private-vlan isolated
SwitchA(config-vlan)#vlan 500
SwitchA(config-vlan)#private-vlan primary
SwitchA(config-vlan)#private-vlan association add 502
```

The configuration is the same as the community VLAN but this time I'm using the **private-vlan isolated** command. Don't forget to **add the association between the primary and secondary VLAN using the private-vlan association add command**. The private-vlan primary command is obsolete because I already did this before, I'm just showing it to keep the configuration complete.

```
SwitchA(config)#interface range fa0/3 - 4
SwitchA(config-if-range)#switchport mode private-vlan host
SwitchA(config-if-range)#switchport private-vlan host-association 500 502
```

This part is exactly the same as the configuration for the community VLAN but I'm configuring interface fa0/3 and fa0/4 which are connected to ComputerC and ComputerD.

```
SwitchA(config)#interface fa0/24
SwitchA(config-if)#switchport mode private-vlan promiscuous
SwitchA(config-if)#switchport private-vlan mapping 500 502
```

I already configured fa0/24 as a promiscuous port but I'm showing it here as well to keep the configuration complete. I do need to create an additional mapping between VLAN 500 (primary) and VLAN 502 (secondary).

Let's verify our work!

```
SwitchA#show interfaces fa0/3 switchport
Name: Fa0/3
Switchport: Enabled
Administrative Mode: private-vlan host
Operational Mode: down
Administrative Trunking Encapsulation: negotiate
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: 500 (VLAN0500) 502 (VLAN0502)
Administrative private-vlan mapping: none
```

Looking good…we can see the host-association between VLAN 500 and 502.

```
SwitchA#show interfaces fastEthernet 0/4 switchport | include host-as
Administrative private-vlan host-association: 500 (VLAN0500) 502 (VLAN0502)
```

A quick look at fa0/4 shows me the same output as fa0/3.

```
SwitchA#show interfaces fa0/24 switchport
Name: Fa0/24
Switchport: Enabled
Administrative Mode: private-vlan promiscuous
Operational Mode: private-vlan promiscuous
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: 500 (VLAN0500) 501 (VLAN0501) 502
(VLAN0502)
```

We can now see that VLAN 501 and VLAN 502 are mapped to primary VLAN 500.

```
SwitchA#show vlan private-vlan

Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------------
-
500     501       community         Fa0/1, Fa0/2, Fa0/24
500     502       isolated          Fa0/3, Fa0/4, Fa0/24
```

Here's a nice clean overview which shows us all the VLANs, the mappings and the interfaces.

```
SwitchA#show vlan private-vlan type

Vlan Type
---- -----------------
500  primary
501  community
502  isolated
```

Or if you only care about the VLAN numbers and the VLAN type this is what you need.

What will the result be of our hard labor?

```
C:\Documents and Settings\ComputerC>ping 192.168.1.254

Pinging 192.168.1.254 with 32 bytes of data:

Reply from 192.168.1.254: bytes=32 time<1ms TTL=128
```

ComputerC can reach the server behind the promiscuous port.

```
C:\Documents and Settings\ComputerD>ping 192.168.1.254

Pinging 192.168.1.254 with 32 bytes of data:

Reply from 192.168.1.254: bytes=32 time<1ms TTL=128
```

ComputerD can also reach the server behind the promiscuous port.

```
C:\Documents and Settings\ComputerC>ping 192.168.1.4

Pinging 192.168.1.4 with 32 bytes of data:

Request timed out.

Ping statistics for 192.168.1.4:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),
```

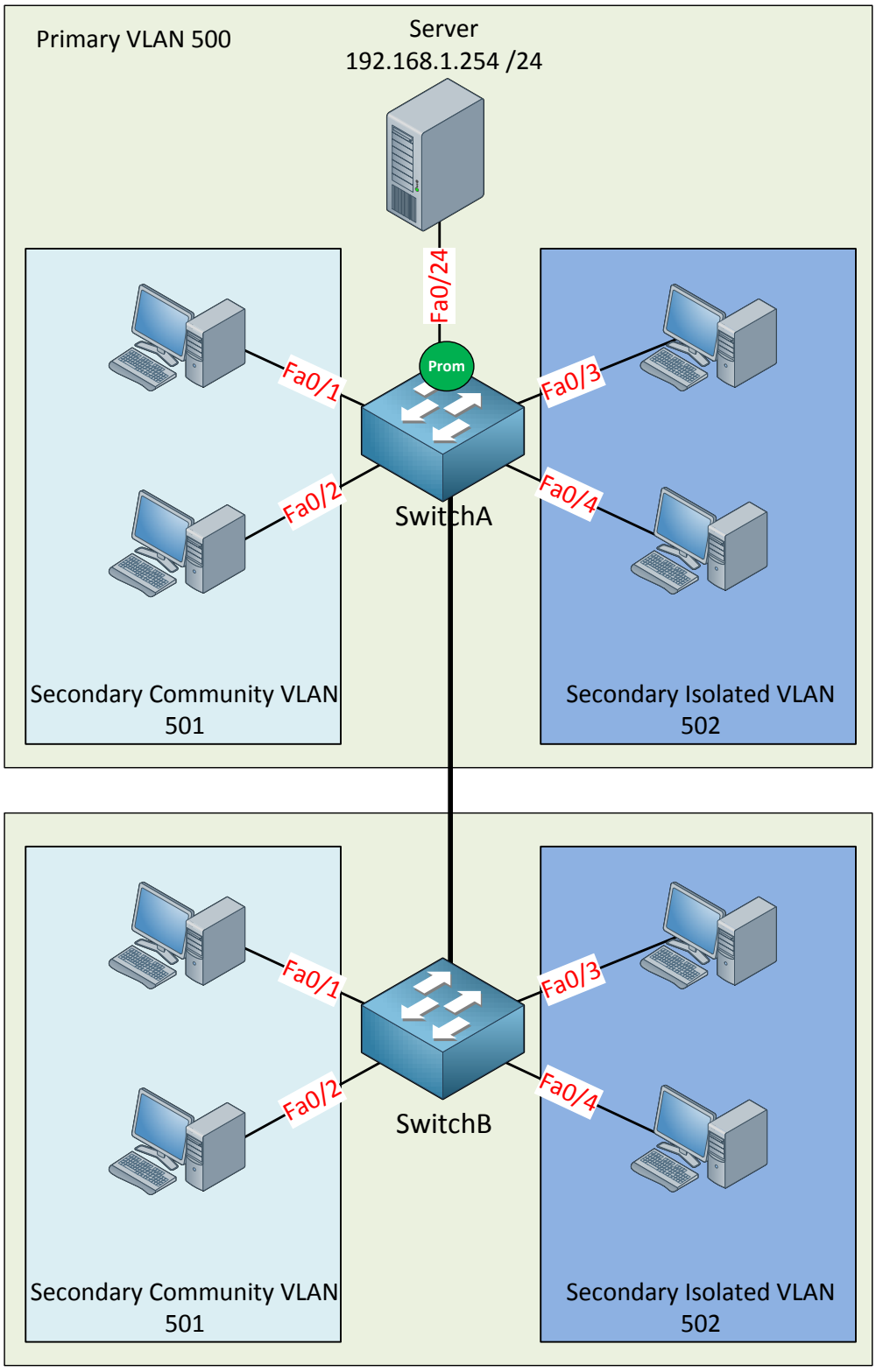
There is no reachability between ComputerC and ComputerD because they are in the isolated VLAN.

What about reachability between VLAN 501 and VLAN 502? Let's give it a try:

```
C:\Documents and Settings\ComputerA>ping 192.168.1.4

Pinging 192.168.1.4 with 32 bytes of data:

Request timed out.

Ping statistics for 192.168.1.4:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),
```

This is ComputerA in VLAN 501 trying to reach ComputerD in VLAN 502. As you can see this isn't possible. You are unable to communicate between different secondary VLANs.

Anything else you need to know about private VLANs?



Private VLANs can be carried over 802.1Q links so it's possible to span your configuration over multiple switches. In the picture above I expanded our configuration to SwitchB. The configuration on SwitchB will be the same as SwitchA. You just need to make sure that VLAN 500, 501 and 502 can be carried over the trunk between SwitchA and SwitchB. Don't forget that because of VTP transparent mode, VLAN information is not synchronized between the two switches. You'll have to create the VLANS yourself on the other switches.

Let me give you a short overview of what we have seen now:

- Devices within a community VLAN can communicate with each other AND the promiscuous port.
- Devices within an isolated VLAN cannot communicate with each other and can ONLY communicate with the promiscuous port.
- The promiscuous port can communicate with any other port.
- Secondary VLANs are unable to communicate with other secondary VLANs.
- Private VLANs can be spanned across multiple switches if you use trunks.

That's all I have for you about private VLANs! What do you think? I hope this all makes sense to you. There are a bunch of different commands you need to use in order to configure this and it may be difficult to remember them all. If you want to try the configuration yourself you can try the following lab:

http://gns3vault.com/Switching/private-vlan.html

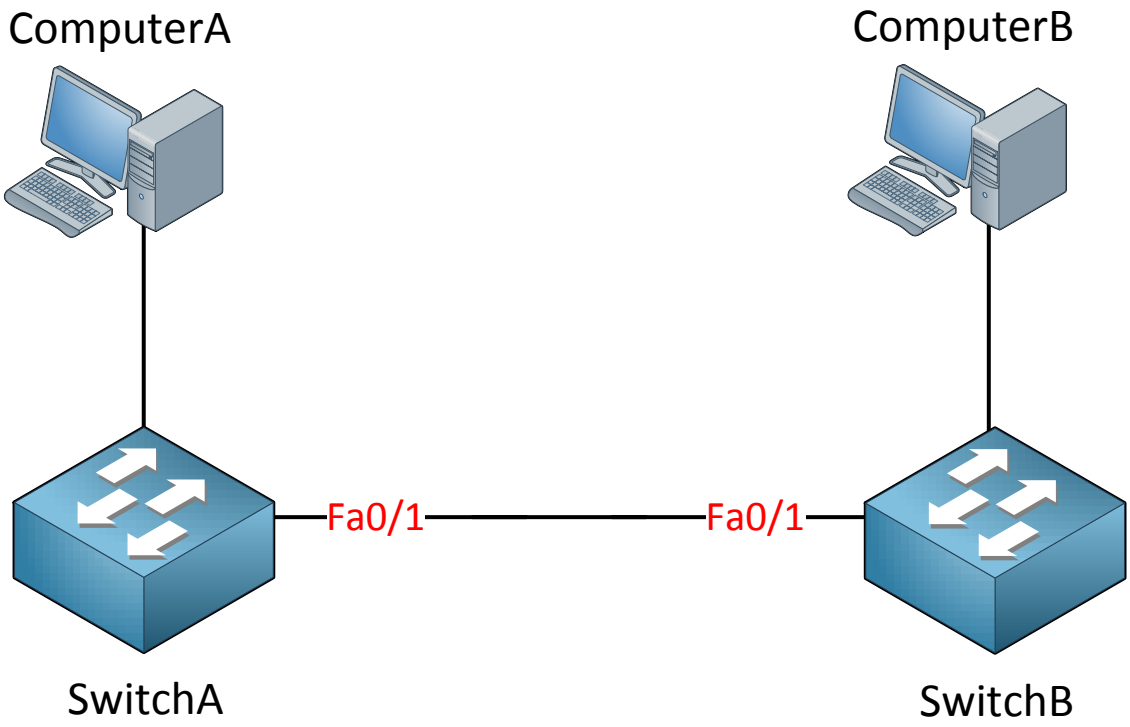
Keep in mind that you **need at least** a Cisco Catalyst 3560 switch to configure private VLANs. If you try this on a Cisco Catalyst 3550 you will notice that some of the commands are accepted but it won't work.

4. STP (Spanning Tree Protocol)

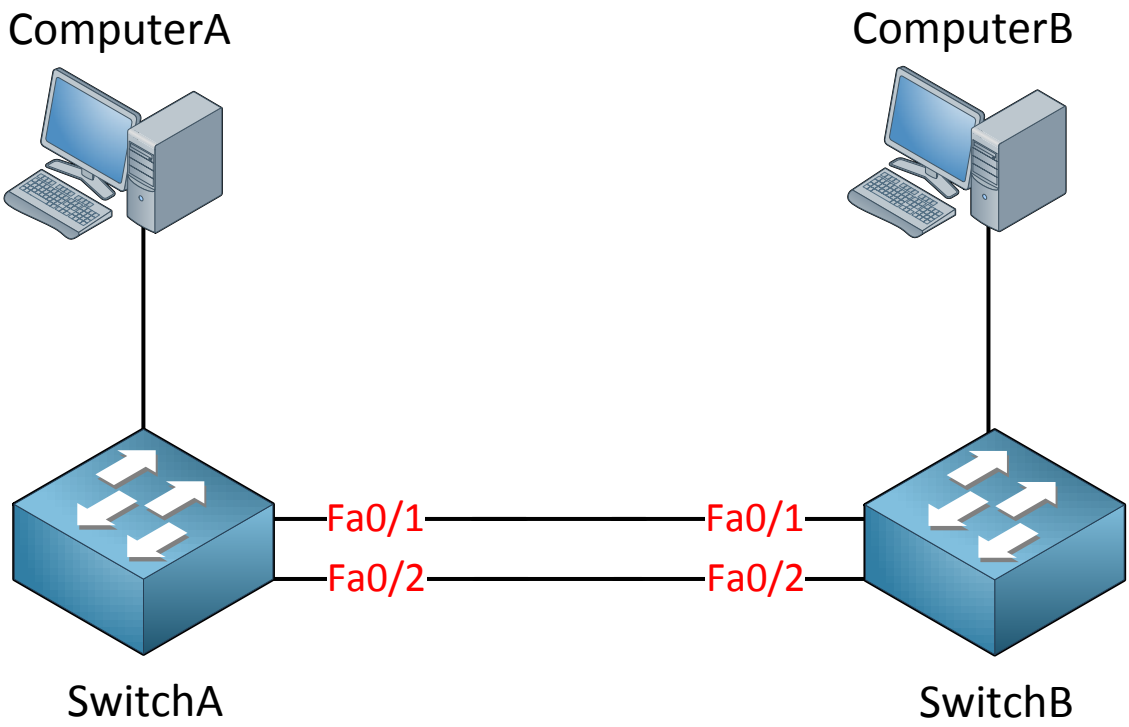
If you studied CCNA you will have a basic understanding of spanning-tree. If you haven't studied CCNA and are still new to switching…don't worry, in the first part of this chapter I will start with the basics of spanning-tree and we'll dive deeper into the material as we move along.

In short, spanning-tree helps you to **create a loop-free topology** in your switched network. The question we should ask ourselves is:

- What causes a loop in a switched network?



We add **loops in our network by adding redundancy** in our switched network. In our picture above we have two switches and only a single link between them. Having a single point of failure isn't something we like to see so to add redundancy we will add another cable.



Very nice…another cable adds redundancy to our network, our single point of failure is gone. However we now have a loop in our network! So why do we have a loop? Let's take a look at the following situation:

1. Computer A sends an ARP request because it's looking for the MAC address of computer B. An ARP request is a broadcast frame.
2. Switch A will forward this broadcast frame on all it interfaces, except the link where the frame originated from.
3. Switch B will receive both broadcast frames.

Now what does switch B do with those broadcast frames?

4. It will forward it out of every link except the interface where it originated from.
5. This means that the frame that was received on Interface fa0/1 will be forwarded on Interface fa0/2.
6. The frame that was received on Interface fa0/2 will be forwarded on Interface fa0/1.

Do you see where this is going? We have a loop! Both switches will keep forwarding over and over until this will happen:

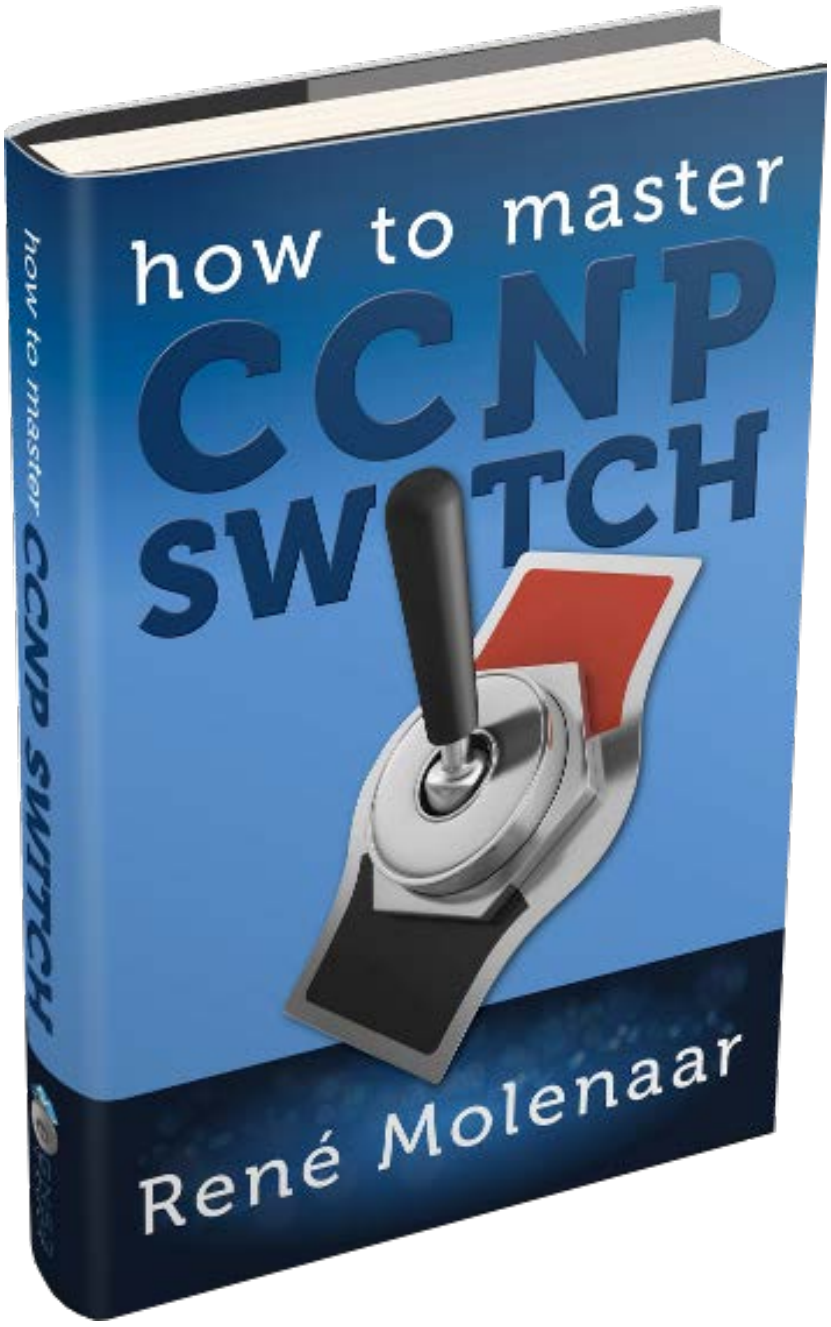
- You fix the loop by disconnecting one of the cables.
- Your switches will crash because they are overburdened with traffic (uh oh!)
- Ethernet frames don't have a TTL (Time to Live) field so frames will loop forever.

The same thing will occur with "unknown unicast traffic". If your switch doesn't know on which interface it can reach a MAC address the frame will be flooded on all interfaces except the one where it originated from.

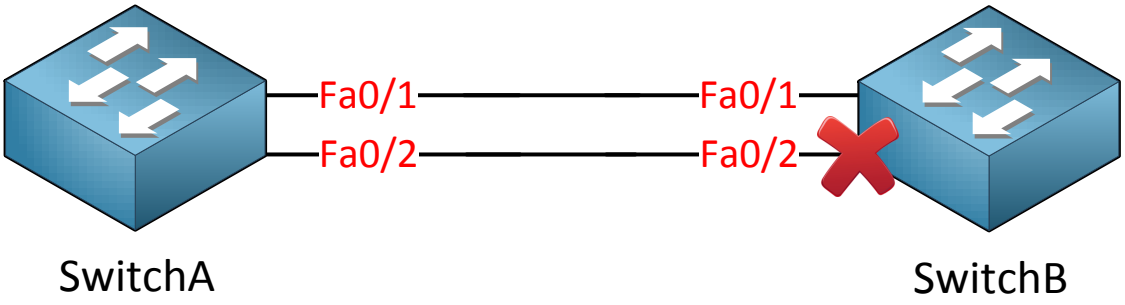
Do you enjoy reading this sample of How to Master CCNP SWITCH ?

Click on the link below to get the full version.

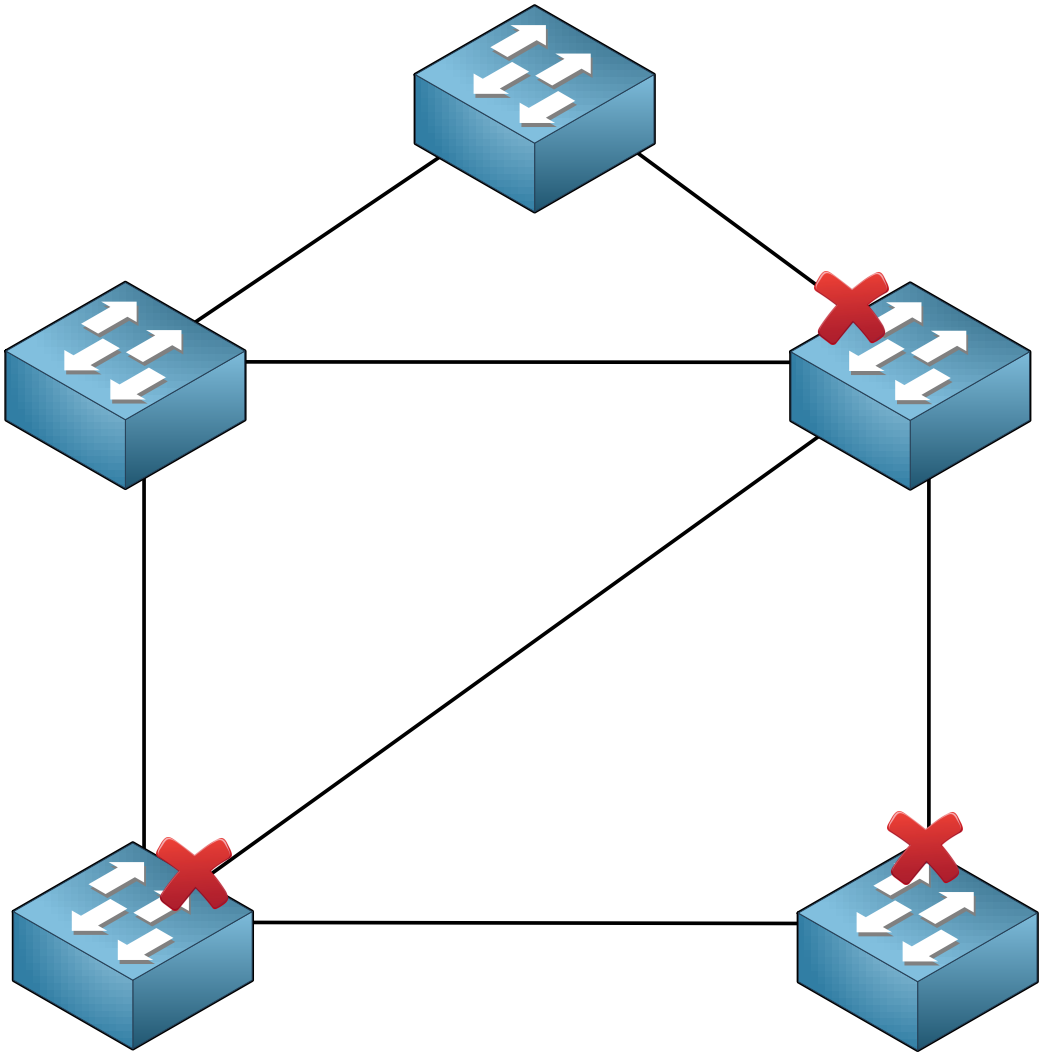
Get How to Master CCNP SWITCH Today



Having a loop in our switched network doesn't sound like a good idea; let's take a look at how spanning-tree works!

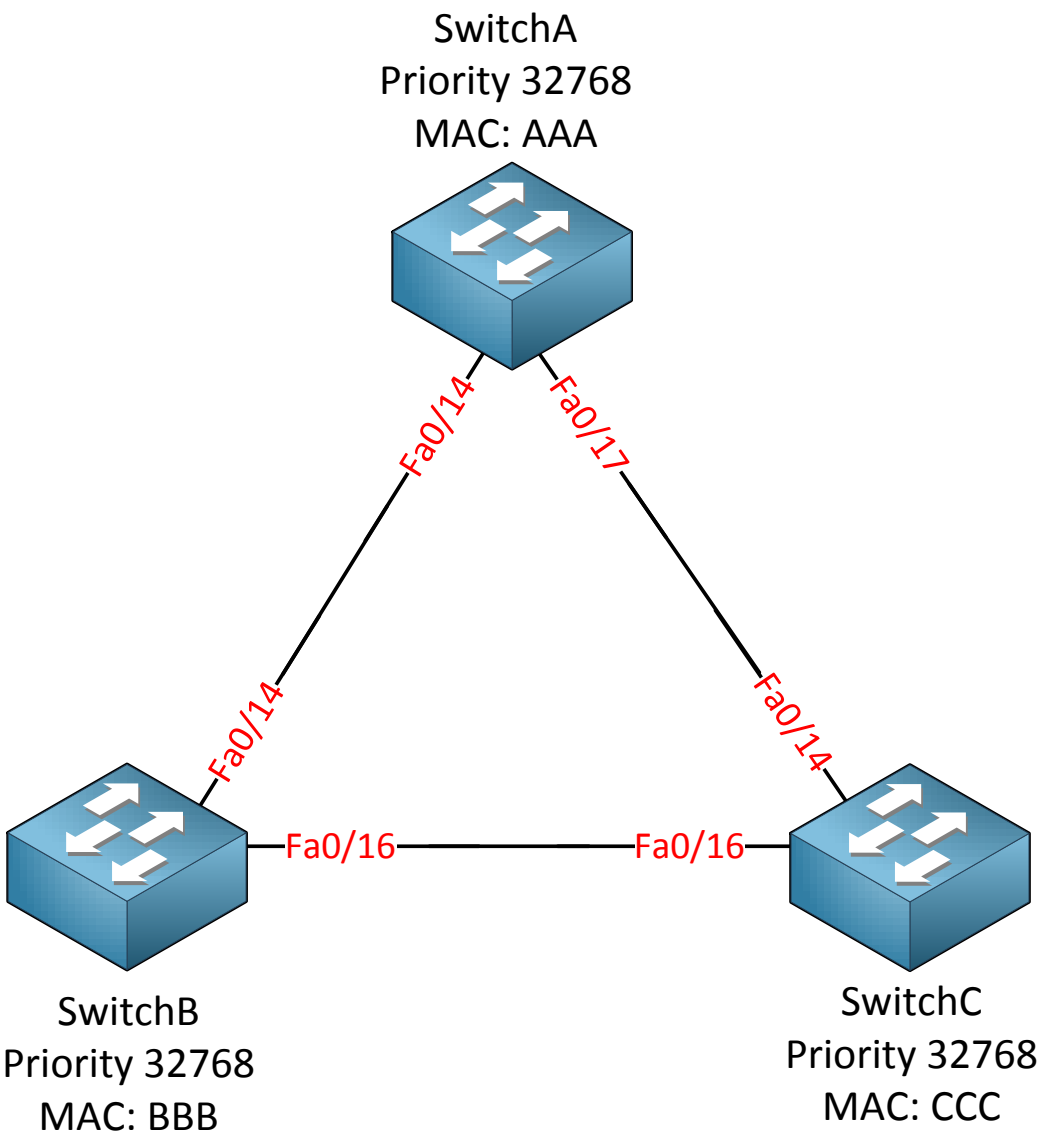


Here is the short answer: spanning-tree will block one or more interfaces in your switched network so the result is a **loop-free topology**.



If you have a (larger) network like the one in the picture above you will see that spanning-tree will block multiple interfaces so that we end up with a loop-free topology.

Now you have an idea how spanning-tree works, let's have a more detailed look to see how it operates.

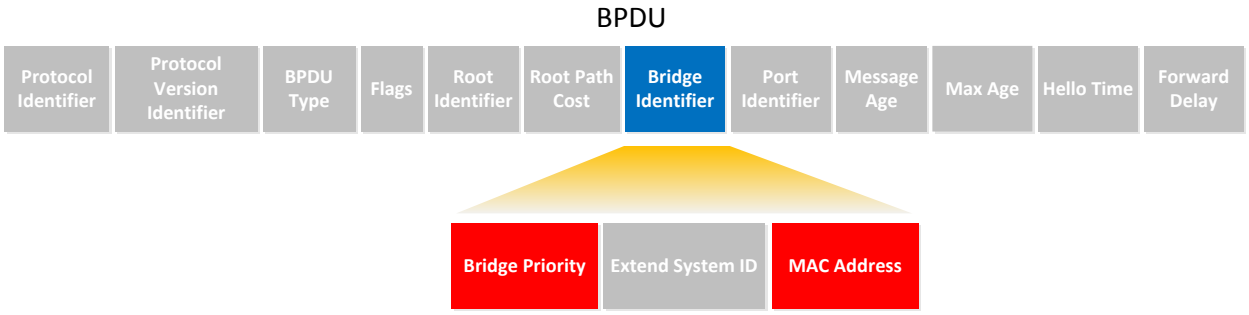


This is the topology I will be using to demonstrate spanning-tree. The switches are connected in a triangle which means that we have redundancy and thus a loop. In the picture you will find the MAC address for each switch but I have simplified them for this example:

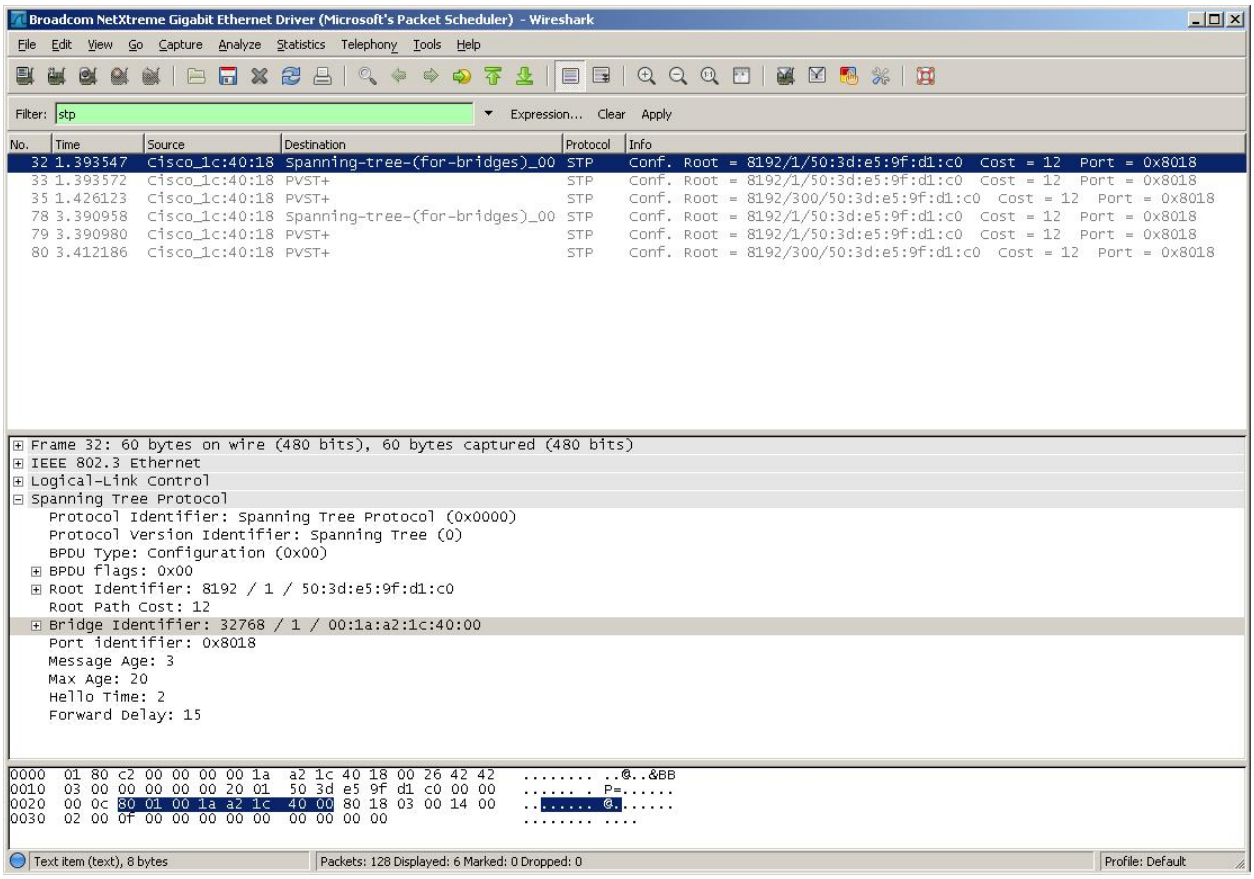
- SwitchA: MAC address AAA
- SwitchB: MAC address BBB
- SwitchC: MAC address CCC

In our picture you can also see a **priority** field which has value 32768 on all switches. This is a default value that we can change if we want, I'll show you later why and how we can do this.

Switches running spanning-tree exchange information with a special message called the **(BPDU) bridge protocol data unit.** All the information in the BPDU is needed to create and maintain the spanning-tree topology.

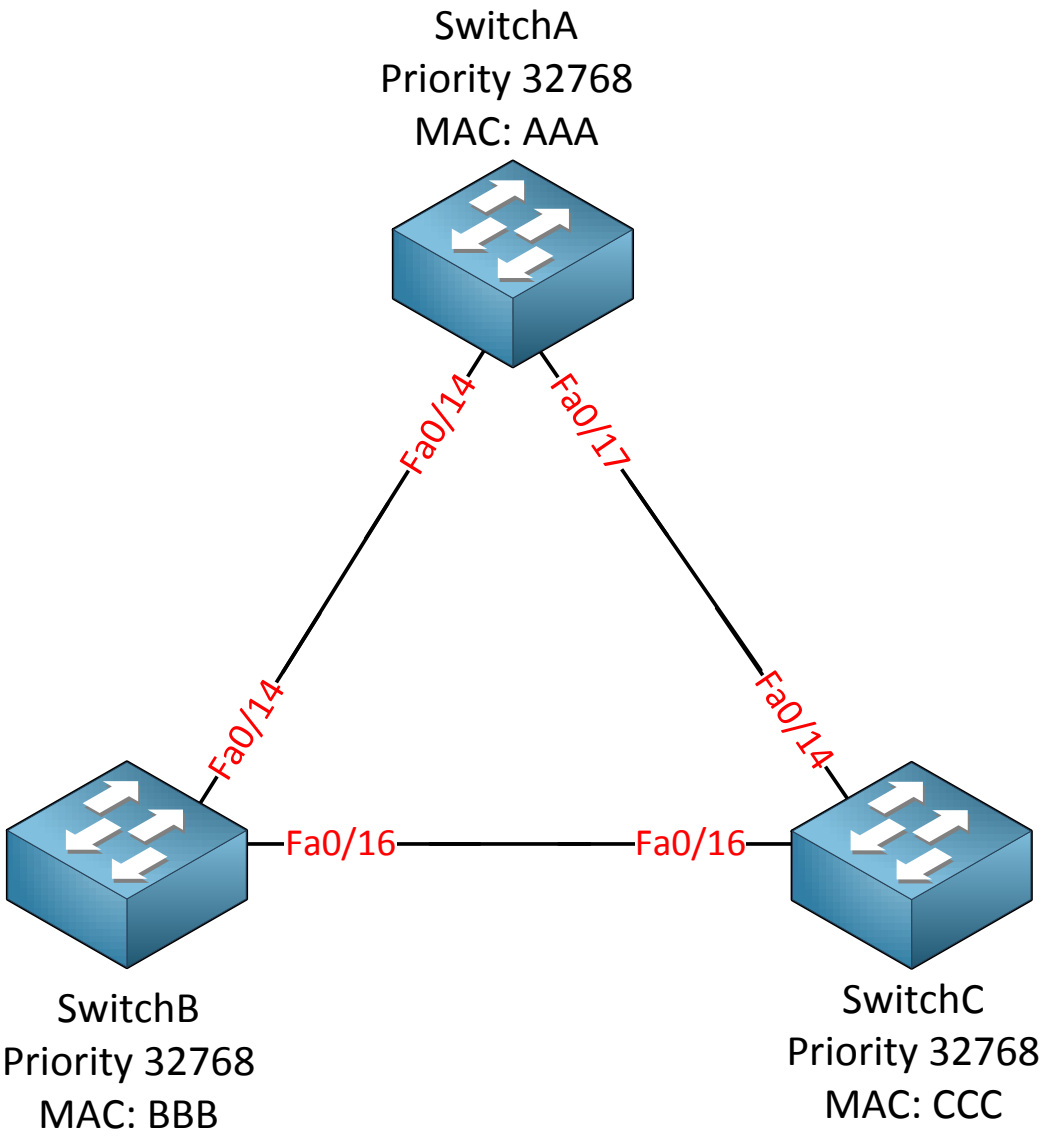


You can see the BPDU in the picture above and the only field that is important right now is the **bridge identifier**. It contains the **bridge priority** and the **MAC address**. It also has a extend system ID field but that's of no concern to us at this moment.

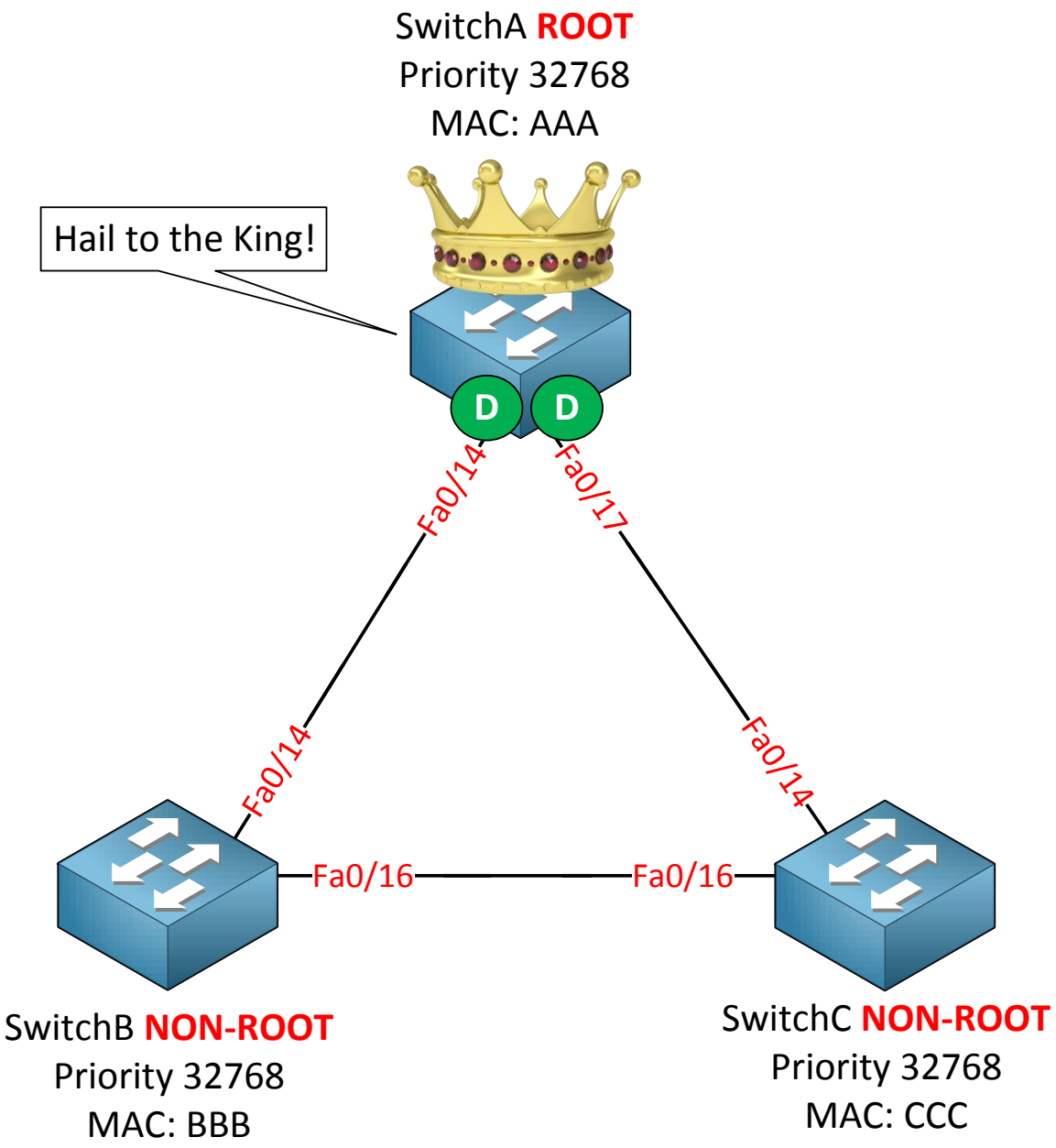


If you run Wireshark on a device that is connected to a Cisco switch you can capture a BPDU and see its contents.

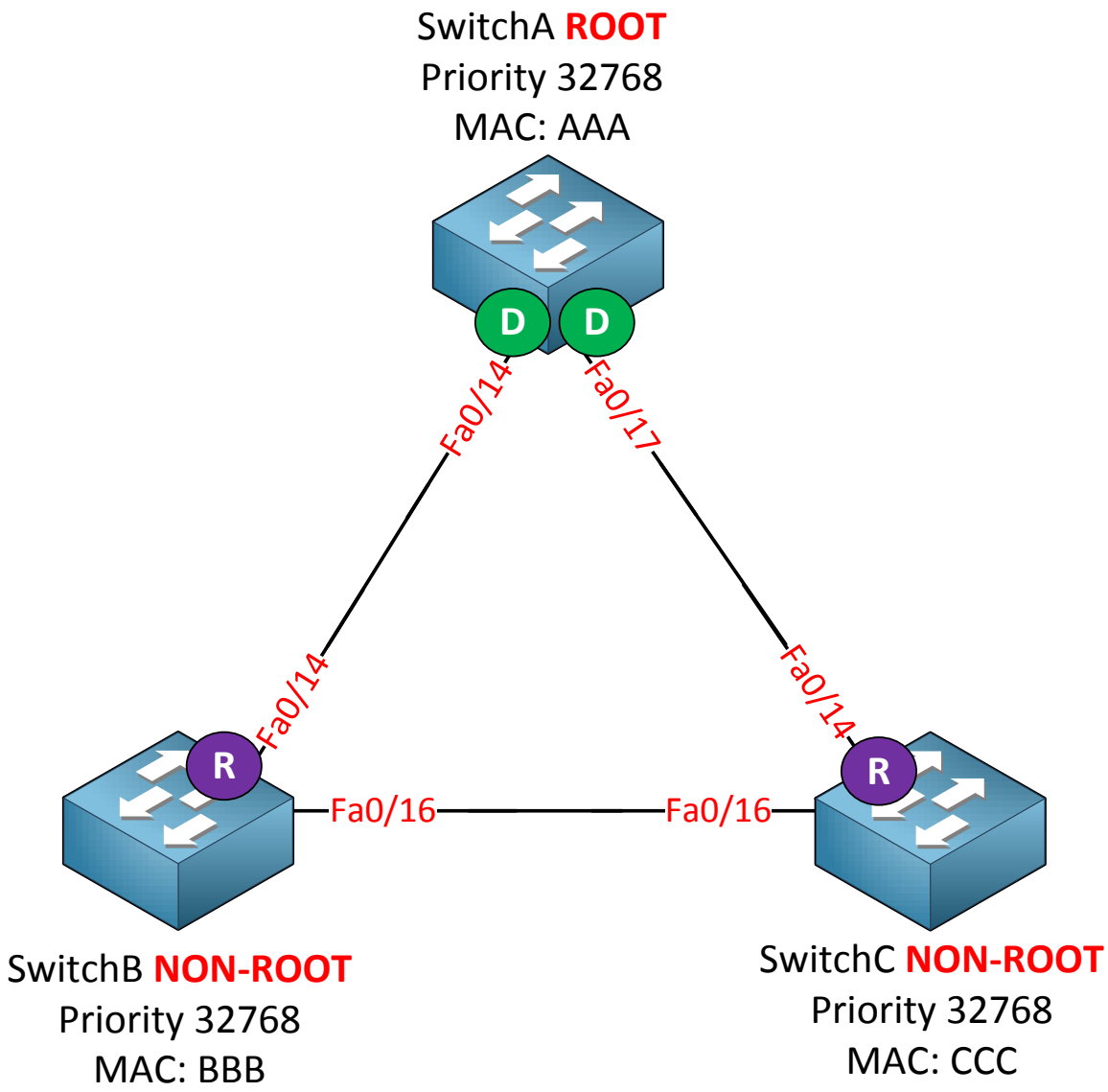
Let me give you a demonstration of how spanning-tree operates and how it uses the information in the bridge identifier:



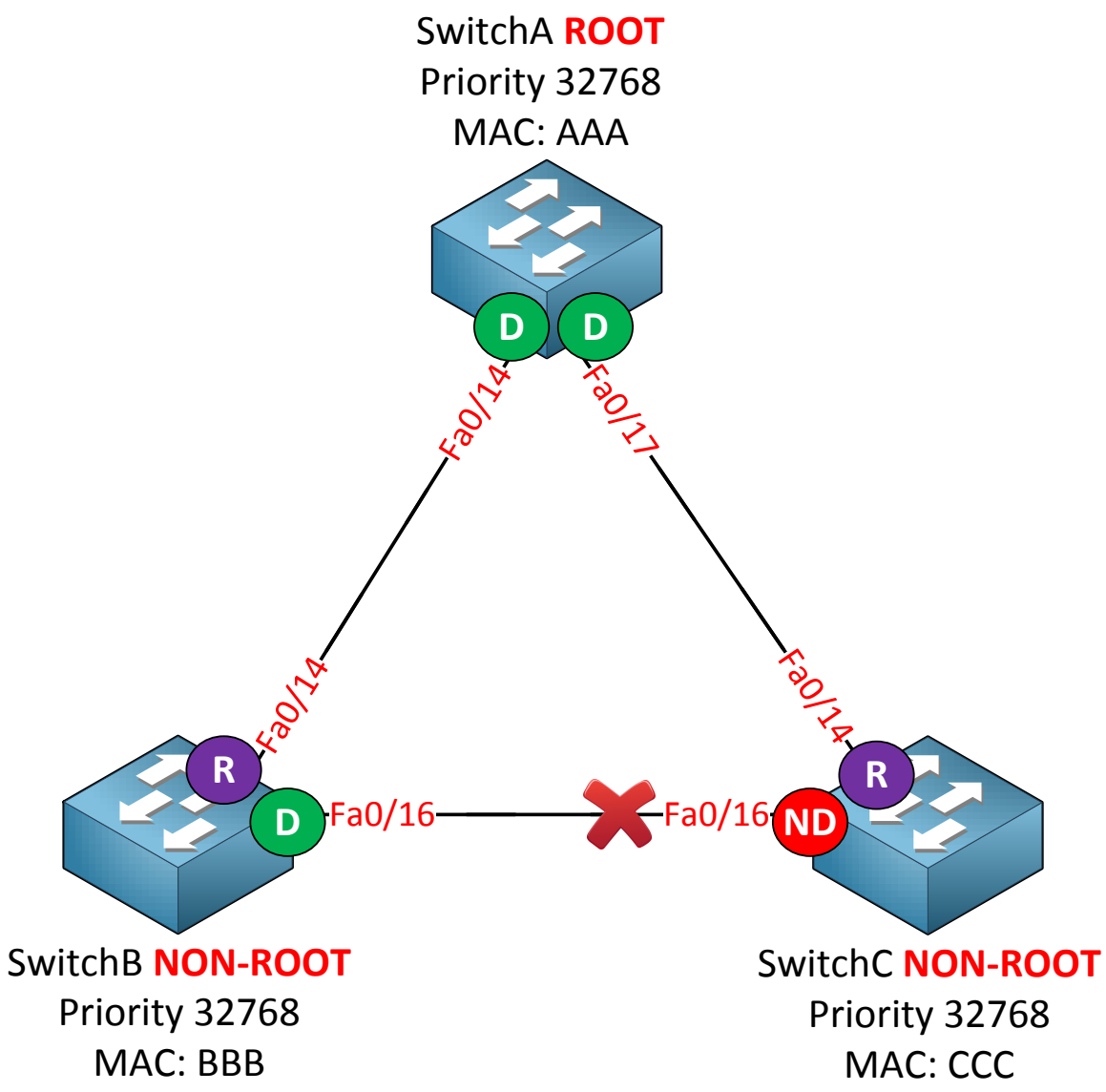
1. The first thing that spanning-tree has to do is **elect a root bridge**. The root bridge is the switch with the lowest **bridge identifier.** The bridge identifier as I just explained consists of the priority plus MAC address. In our example SwitchA will become the root bridge. The priority is the same on all switches so the MAC address will be the tiebreaker!



2. SwitchA is now the root bridge because it has the best bridge identifier. All the other switches are called **non-root**. Interfaces that forward traffic are called **designated ports** in spanning-tree. On a root bridge the interfaces are always in forwarding mode because the non-root switches will need to find the root bridge. In the picture above I added the "D" to show that the fa0/14 and fa0/17 interfaces on SwitchA are designated and forwarding traffic.



3. All the non-root switches have to find the **shortest path to the root bridge**. So what is the shortest path? Spanning-tree is smart enough to decide that a Gigabit interface is a better choice than a FastEthernet link. To keep things simple at this stage I am using FastEthernet links between all switches. SwitchB its fa0/14 interface is the shortest path to get to the root bridge. SwitchC its fa0/14 is also the shortest path to get to the root bridge. The interface that leads us to the root bridge is called the **root port** and is forwarding traffic.



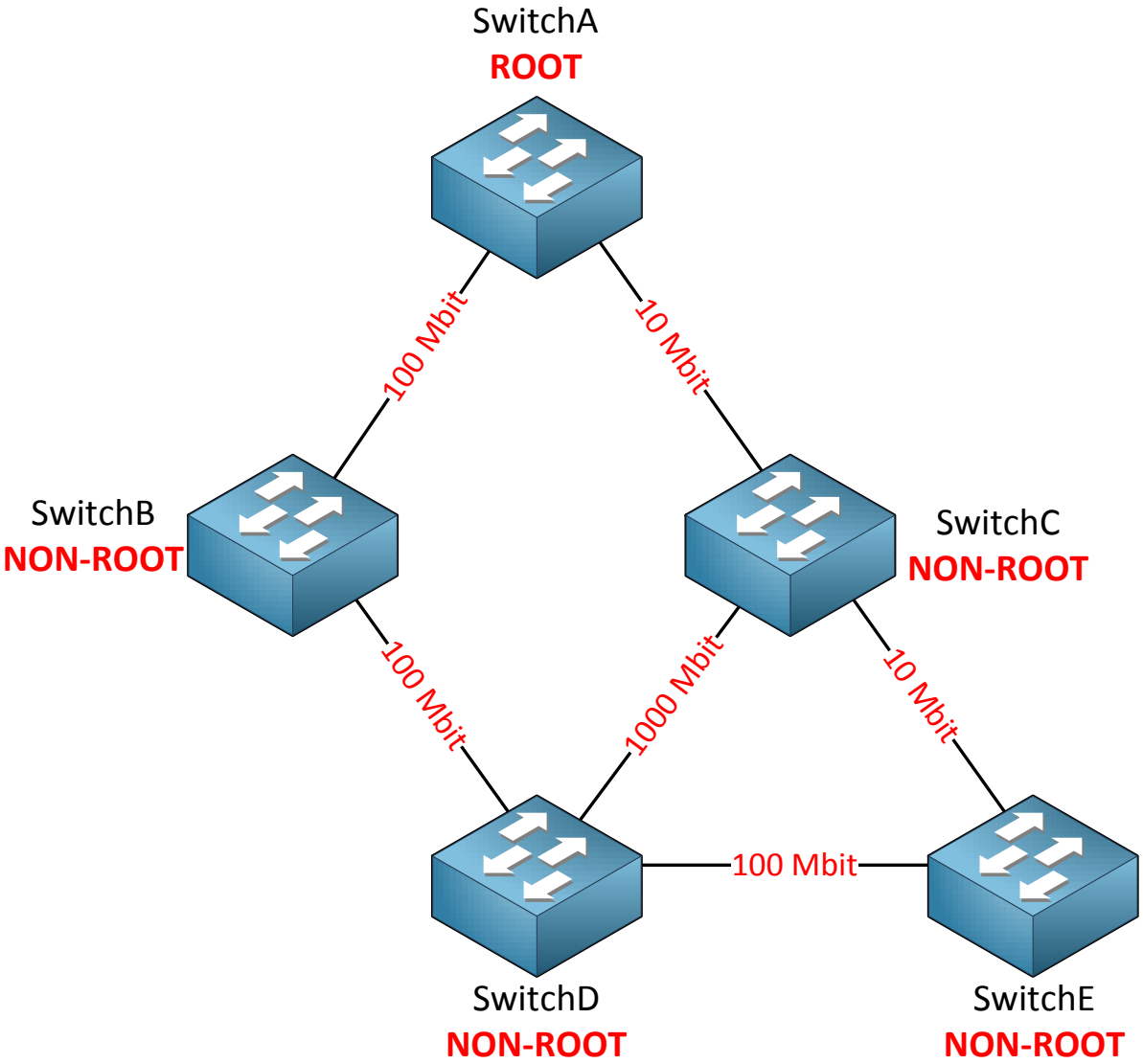
4. In order to break the loop we have to block an interface between SwitchB and SwitchC. So which one are we going to block? SwitchB and SwitchC will duke it out by comparing their bridge identifier. Keep in mind the bridge identifier consists of the priority and MAC address. The lowest bridge identifier is the best one, SwitchB and SwitchC have the same priority but SwitchB has a lower MAC address. SwitchB will win this battle and as a result the fa0/16 of SwitchC will be blocked. A port that is **blocking** traffic is called a **non-designated** port. The fa0/16 interface of SwitchB will become a designated port.

Because the priority is 37268 by default the MAC address is the tie-breaker for the root bridge election. Which switch do you think will become the root bridge?

Your brand-spanking-brand-new-just-out-of-the-box switch or that old dust collector that has been in the datacenter for 10 years? The old switch probably has a lower MAC address and will become the root bridge…not a good idea right? I'll show you how to change the priority so the MAC address is no longer the tie-breaker!

Are you following me so far? Good! You just learned the basics of spanning-tree. Let's add some more detail to this story…

Non-root bridges need to find the **shortest path to the root bridge**. In our previous example this was easy because all the interfaces are FastEthernet. What will happen if we have a mix of different interface types like Ethernet, FastEthernet and Gigabit? Let's find out!



In the picture above we have a larger network with multiple switches. You can also see that there are different interface types, we have Ethernet (10 Mbit), FastEthernet (100Mbit) and Gigabit (1000Mbit). SwitchA on top is the root bridge so all other switches are non-root and need to find the shortest path to the root bridge.

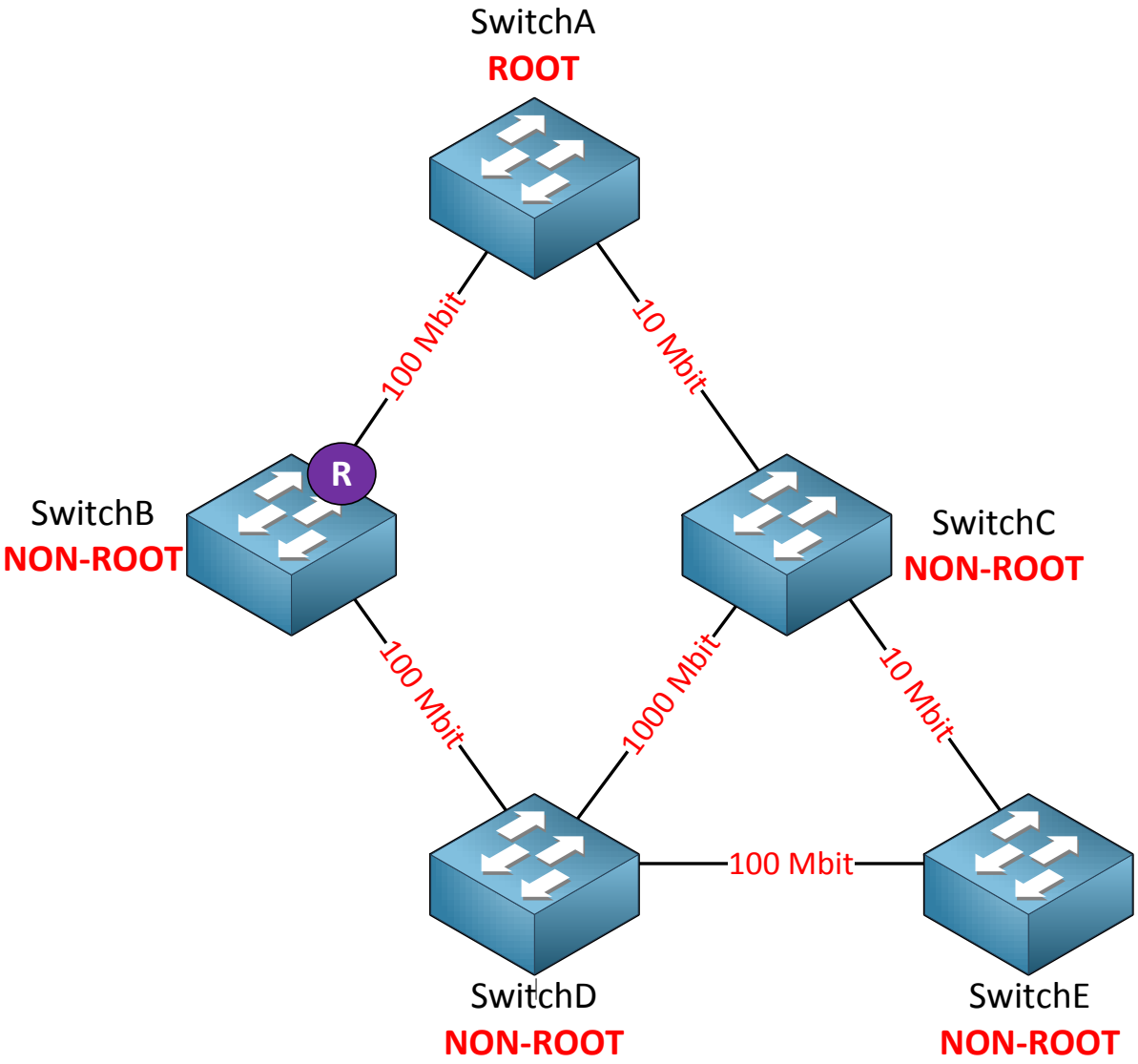
	Cost
10 Mbit	100
100 Mbit	19
1000 Mbit	4

Spanning-tree uses **cost** to determine the shortest path to the root bridge. The slower the interface, the higher the cost is. The path with the lowest cost will be used to reach the root bridge.

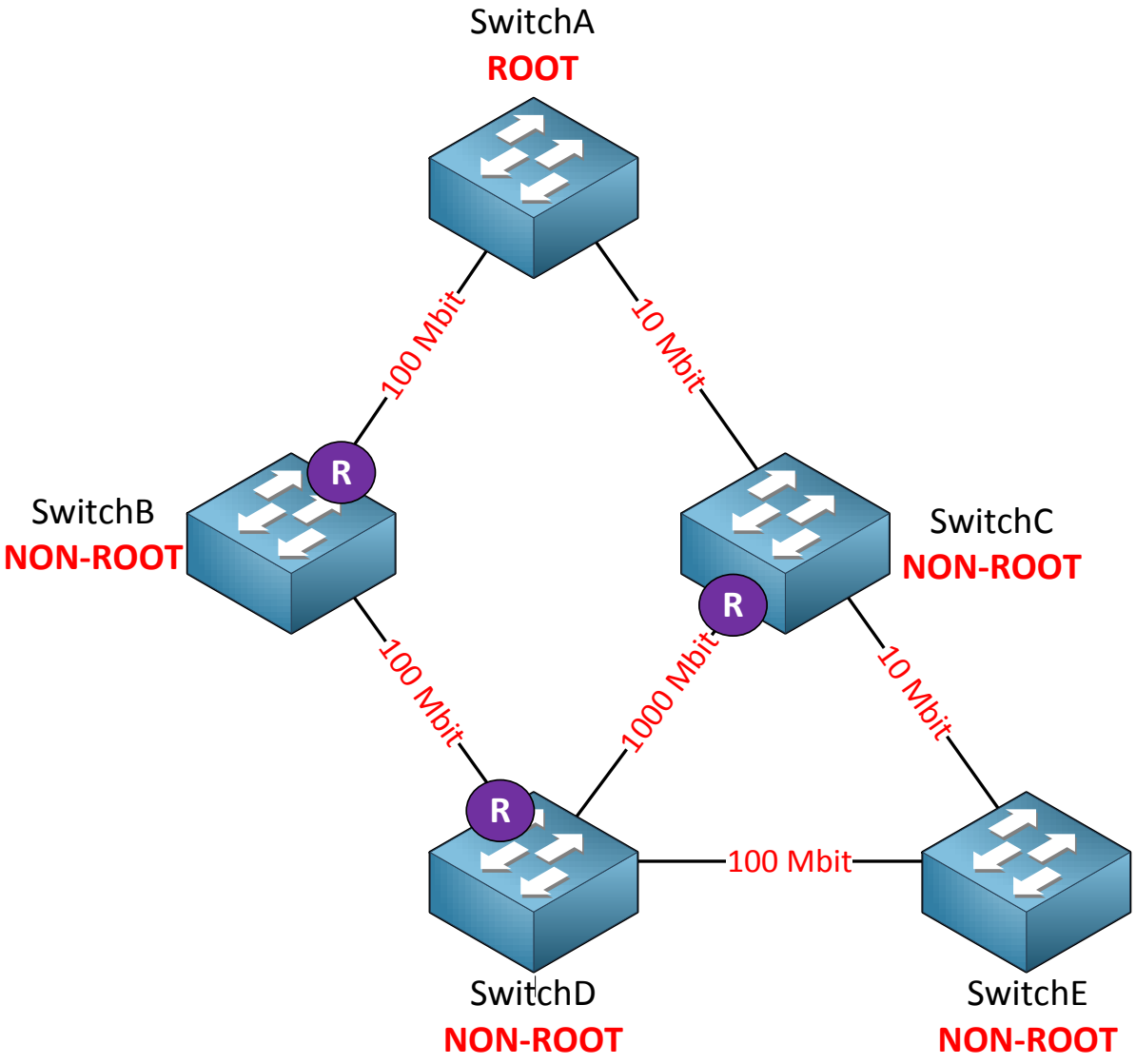
BPDU

In the BPDU you can see a field called **root path cost**. This is where each switch will insert the **cost of its shortest path** to the root bridge. Once the switches found out which switch is declared as root bridge they will look for the shortest path to get there. **BPDUs will flow from the root bridge downwards to all switches**.

If you studied CCNA or CCNP ROUTE then this story about spanning-tree cost might sound familiar. OSPF (Open Shortest Path First) also uses cost to calculate the shortest path to its destination. Both spanning-tree and OSPF use cost to find the shortest path but there is one big difference. OSPF builds a topology database (LSDB) so all routers know exactly what the network looks like. Spanning-tree is "dumb"…switches have no idea what the topology looks like. BPDUs flow from the root bridge downwards to all switches, switches will make a decision based on the BPDUs that they receive!

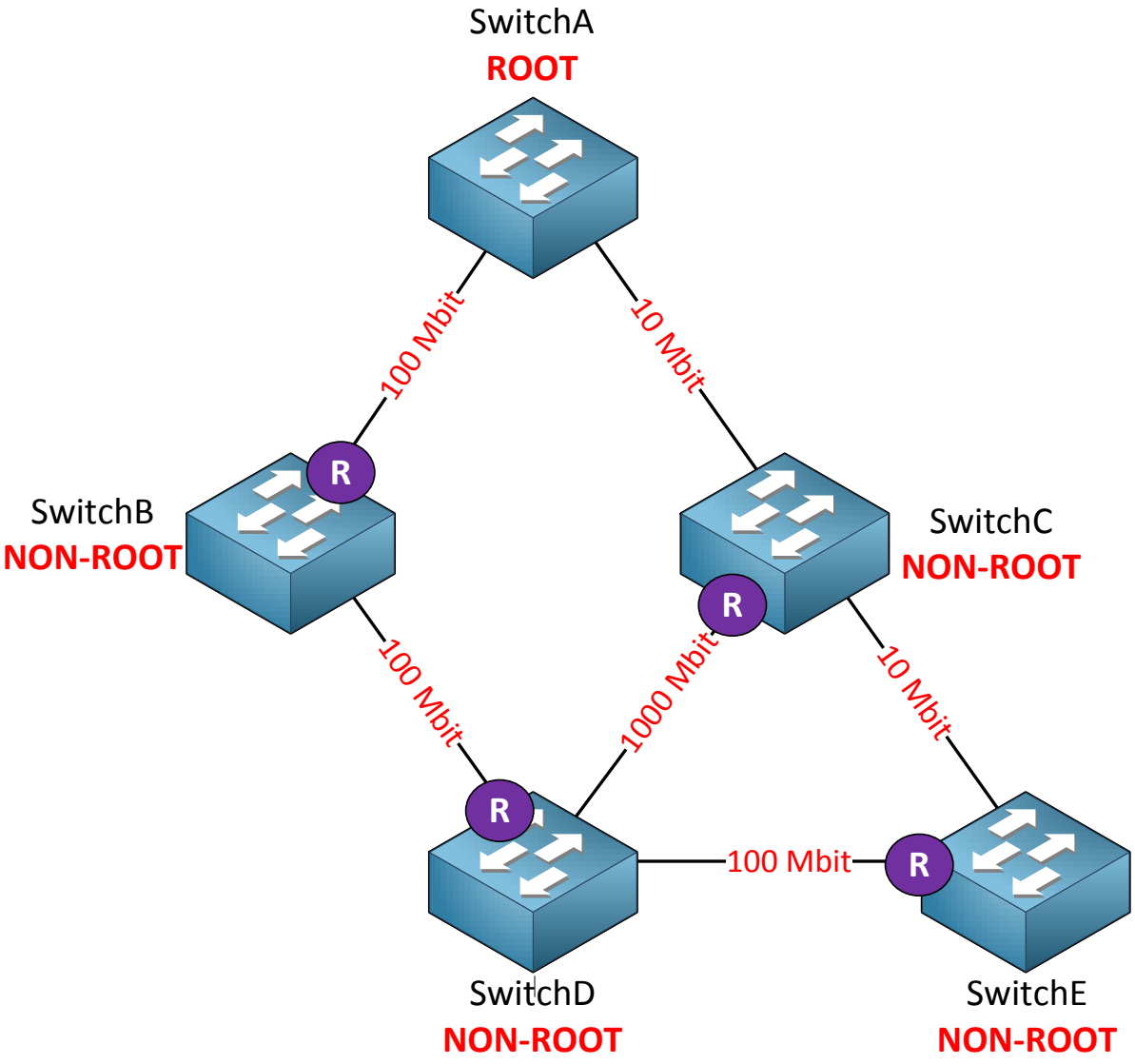


SwitchB will use the direct link to SwitchA as its root port since this is a 100 Mbit interface and has a cost of 19. It will forward BPDUs towards SwitchD; in the root path cost field of the BPDU you will find a cost of 19.



This picture needs some more explanation so let me break it down:

- SwitchC will receive BPDUs on its 10 Mbit interface (cost 100) and on its 1000 Mbit interface (cost 4). It will use its 1000 Mbit interface as its root port.
- SwitchC will forward BPDUs to SwitchD. The root path cost field will be 100.
- SwitchD receives a BPDU from SwitchB with a root path cost of 19.
- SwitchD receives a BPDU from SwitchC with a root path cost of 100.
- The path through SwitchB is shorter so this will become the root port for SwitchD.
- SwitchD will forward BPDUs towards SwitchC and SwitchE. In the root path cost field of the BPDU we will find a cost of 38 (its root path cost of 19 + its own interface cost of 19).
- SwitchC will forward BPDUs towards SwitchE and inserts a cost of 42 in the root path cost field (19 + 19 + 4).

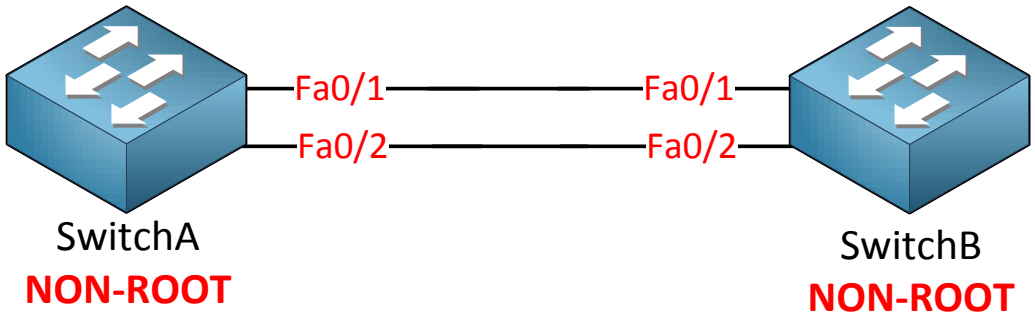


SwitchE receives BPDUs from SwitchC and SwitchD. In the BPDU we will look at the root path cost field and we'll see the following information:

- BPDU from SwitchC: cost 42
- BPDU from SwitchD: cost 38

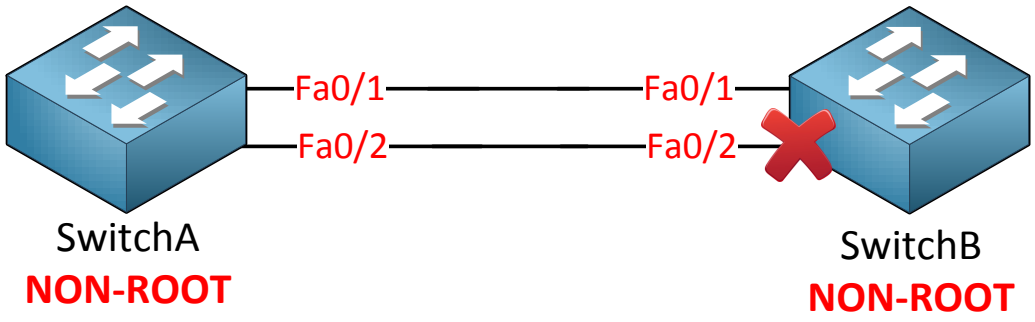
SwitchE will add the cost of its own interface towards SwitchD so the total cost to reach the root bridge through SwitchD is 38 + 19 (cost of 100 Mbit interface) = 57. The total cost to reach the root bridge through SwitchC is 42 + 100 (10 Mbit interface) = 142. As result it will select the interface towards SwitchD as its root port.

Are you following me so far? Keep in mind that switches only make decisions on the BPDUs that they receive! They have no idea what the topology looks like. The only thing they do know is on which interface they received the **best BPDU.** The best BPDU is the one with the shortest path to the root bridge!



What if the cost is equal? Take a look at the picture above. SwitchA is the root bridge and SwitchB is non-root. We have two links between these switches so that we have redundancy. Redundancy means loops so spanning-tree is going to block one the interfaces on SwitchB.

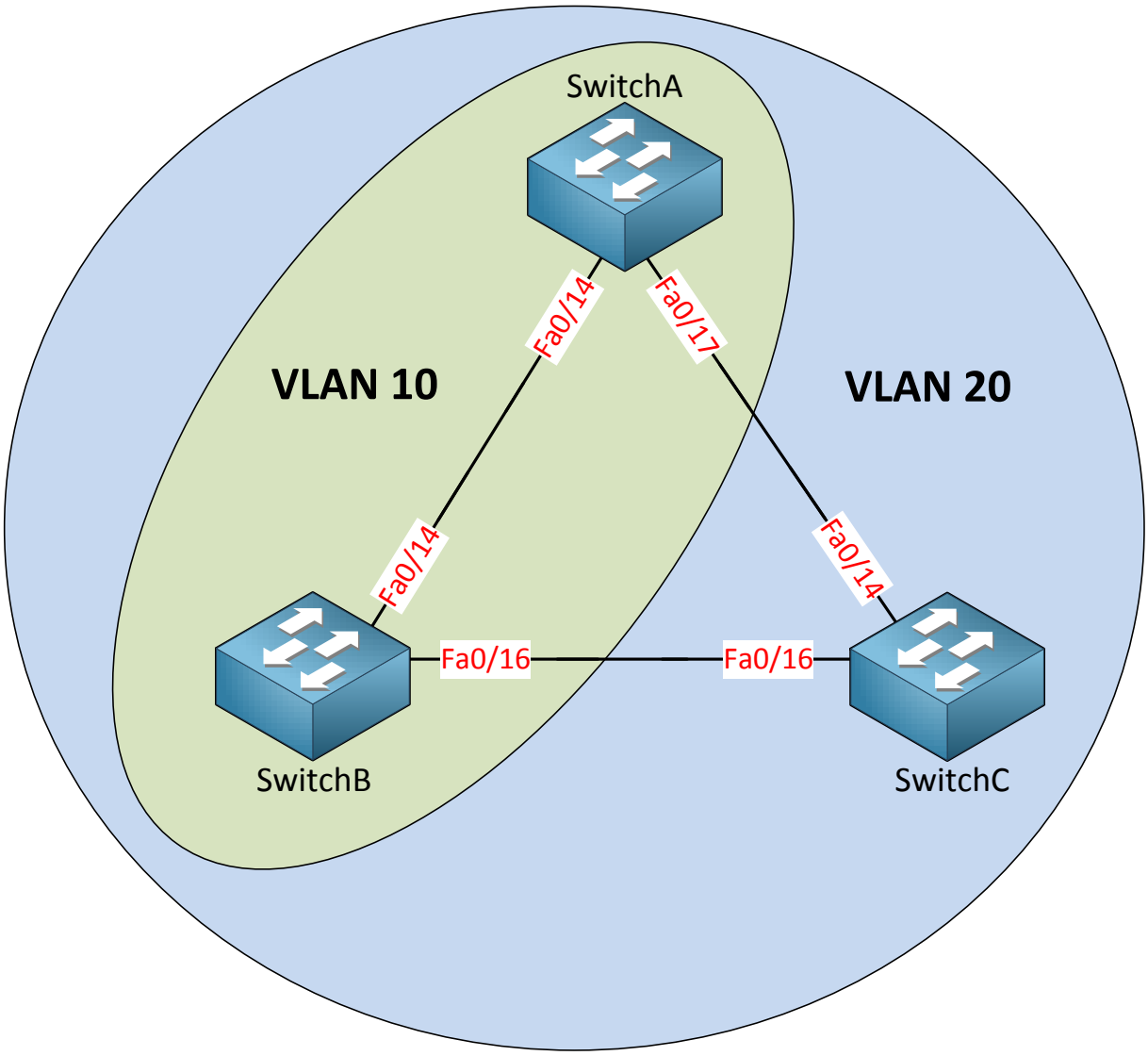
SwitchB will receive BPDUs on both interfaces but the root path cost field will be the same! Which one are we going to block? Fa0/1 or fa0/2?



When the cost is equal spanning-tree will look at the **port priority.** By default the port priority is the **same for all interfaces** which means that the **interface number will be the tie-breaker.** The lowest interface number will be chosen so fa0/2 will be blocked here. Of course port priority is a value that we can change so we can choose which interface will be blocked, I'll show you later how to do this!

Whenever spanning-tree has to make a decision, this is the list that it will use. This is something to write down and remember:

1) **Lowest bridge ID**: the switch with the lowest bridge ID becomes the root bridge.
2) **Lowest path cost to root bridge**: when the switch receives multiple BPDUs it will select the interface that has the lowest cost to reach the root bridge as the root port.
3) **Lowest sender bridge ID**: when a switch is connected to two switches that it can use to reach the root bridge and the cost to reach the root bridge is the same, it will select the interface connecting to the switch with the lowest bridge ID as the root port.
4) **Lowest sender port ID**: when the switch has two interfaces connecting to the same switch, and the cost to reach the root bridge is the same it will use the interface with the lowest number as the root port.



There's more to spanning-tree! Take a look at the picture above:

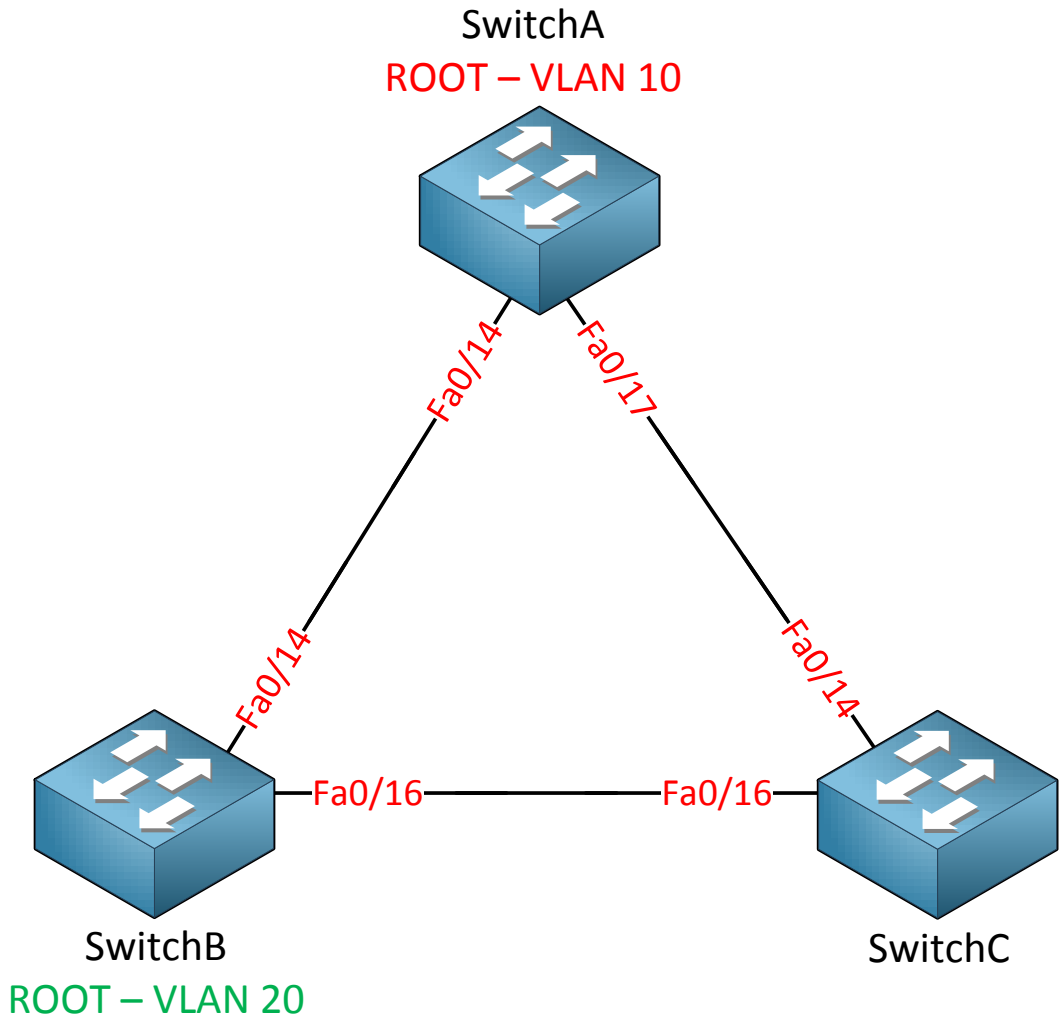
- VLAN 10 is configured on SwitchA and SwitchB.
- VLAN 20 is configured on SwitchA, SwitchB and SwitchC.

Question for you: do we have a loop in VLAN 10? What about VLAN 20?

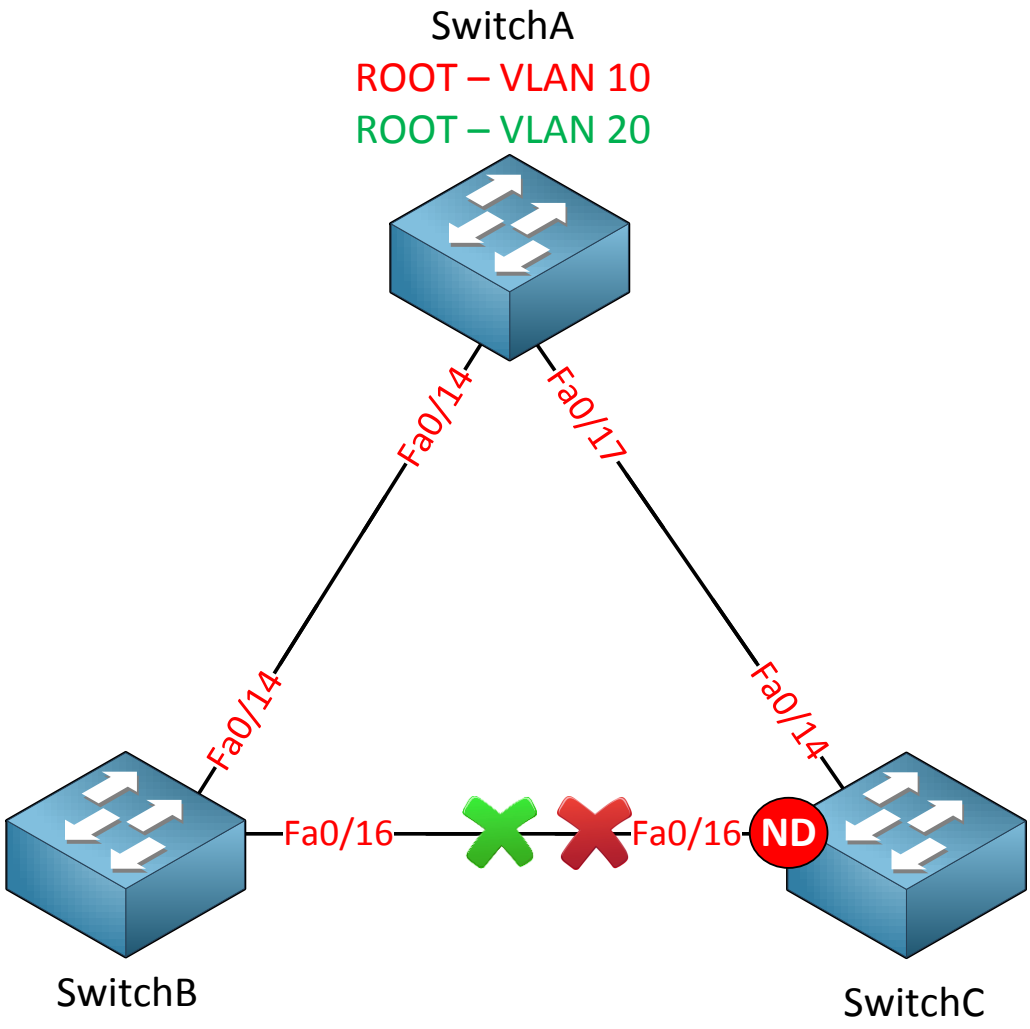
There's a big difference between our **physical** and **logical** topology. We don't have a loop in VLAN 10 because it only runs on the link between SwitchA and SwitchB. We DO have a loop within VLAN 20 however.

How does spanning-tree deal with this? Simple…we'll just calculate a different spanning-tree for each VLAN! The oldest version of spanning-tree is called **CST (Common Spanning-Tree)** and is defined in the 802.1D standard. It only calculates a **single spanning-tree for all VLANs.**

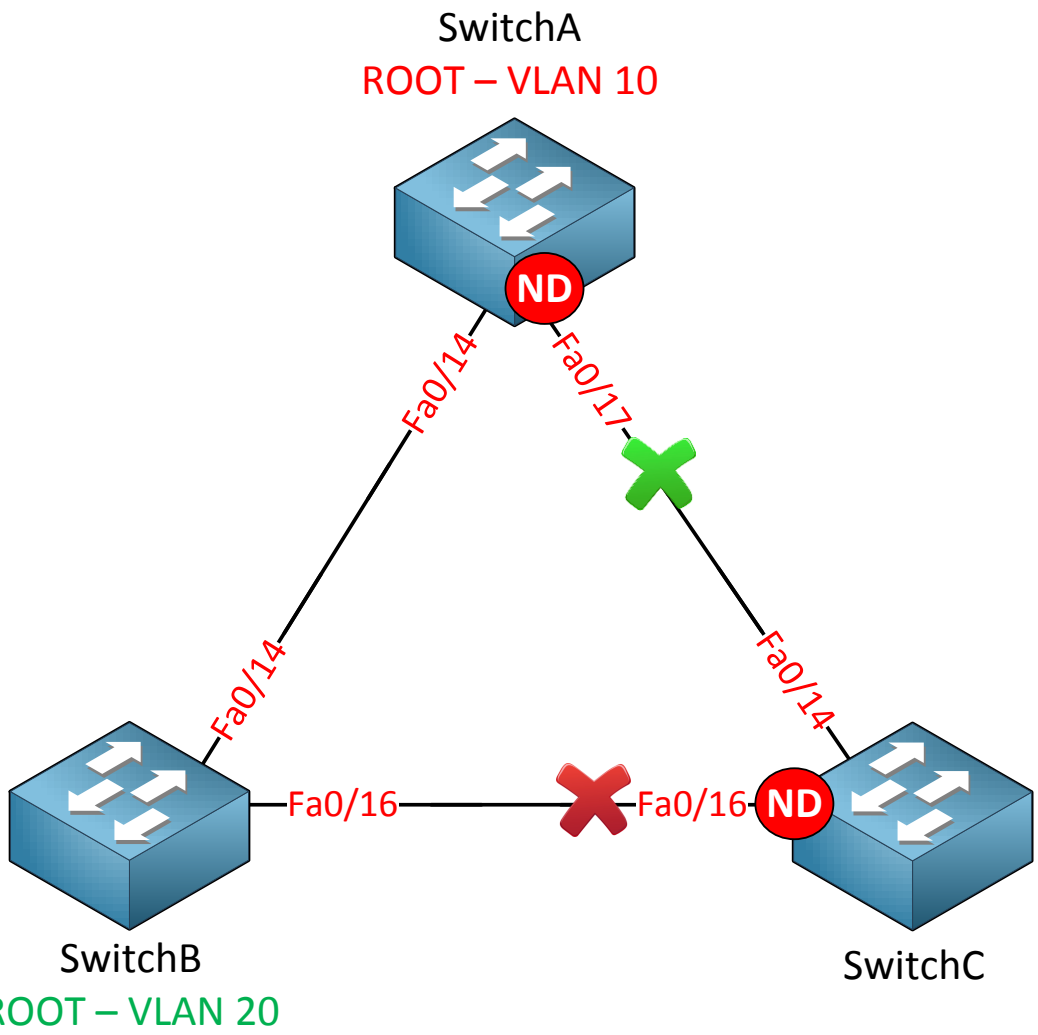
Another version of spanning-tree is able to calculate a topology for **each VLAN**. This version is called **PVST (Per VLAN Spanning-Tree)** and it's the **default on Cisco switches.**



If we use PVST we can create a different root bridge for each VLAN if we want. SwitchA could be the root bridge for VLAN 10 and SwitchB could be the root bridge for VLAN 20. Why would you want to do this?



If I would make one switch root bridge for both VLANs then one interface will be blocked for both VLANs. In my example above SwitchA is the root bridge for VLAN 10 and 20 and as a result the fa0/16 interface on SwitchC is blocked for **both VLANs.** No traffic will be forwarded on the fa0/16 interface at all. Imagine these were 10 Gigabit interfaces. It would be a shame if one of those expensive interfaces wasn't doing anything right?



If I choose another switch as the root bridge for VLAN 20 we will see different results. In my example I made SwitchB the root bridge for VLAN 20. As you can see the fa0/16 interface on SwitchB is blocked for VLAN 10 while the fa0/16 interface on SwitchA is blocked for VLAN 20. The advantage of having multiple root bridges is that I can do some **load sharing/balancing.**

How are we doing so far? There's one more topic I want to discuss before we dive into the configuration of spanning-tree. We'll take a look at the timers of spanning-tree.

If you have played with some Cisco switches before you might have noticed that every time you plug in a cable the led above the interface was orange and after a while became green. What is happening at this moment is that spanning tree is determining the state of the interface.

This is what happens as soon as you plug in a cable:

- **Listening state**: Only a root or designated port will move to the listening state. The non-designated port will stay in the blocking state.No data transmission occurs at this state for 15 seconds just to make sure the topology doesn't change in the meantime. After the listening state we move to the learning state.

- **Learning state:** At this moment the interface will process Ethernet frames by looking at the source MAC address to fill the mac-address-table. Ethernet frames however are not forwarded to the destination. It takes 15 seconds to move to the next state called the forwarding state.

- **Forwarding state:** This is the final state of the interface and finally the interface will forward Ethernet frames so that we have data transmission!

When a port is not a designated or root port it will be in **blocking mode**.

This means it takes 30 seconds in total to move from listening to forwarding…that's not really fast right? This will happen on **all interfaces** on the switch.

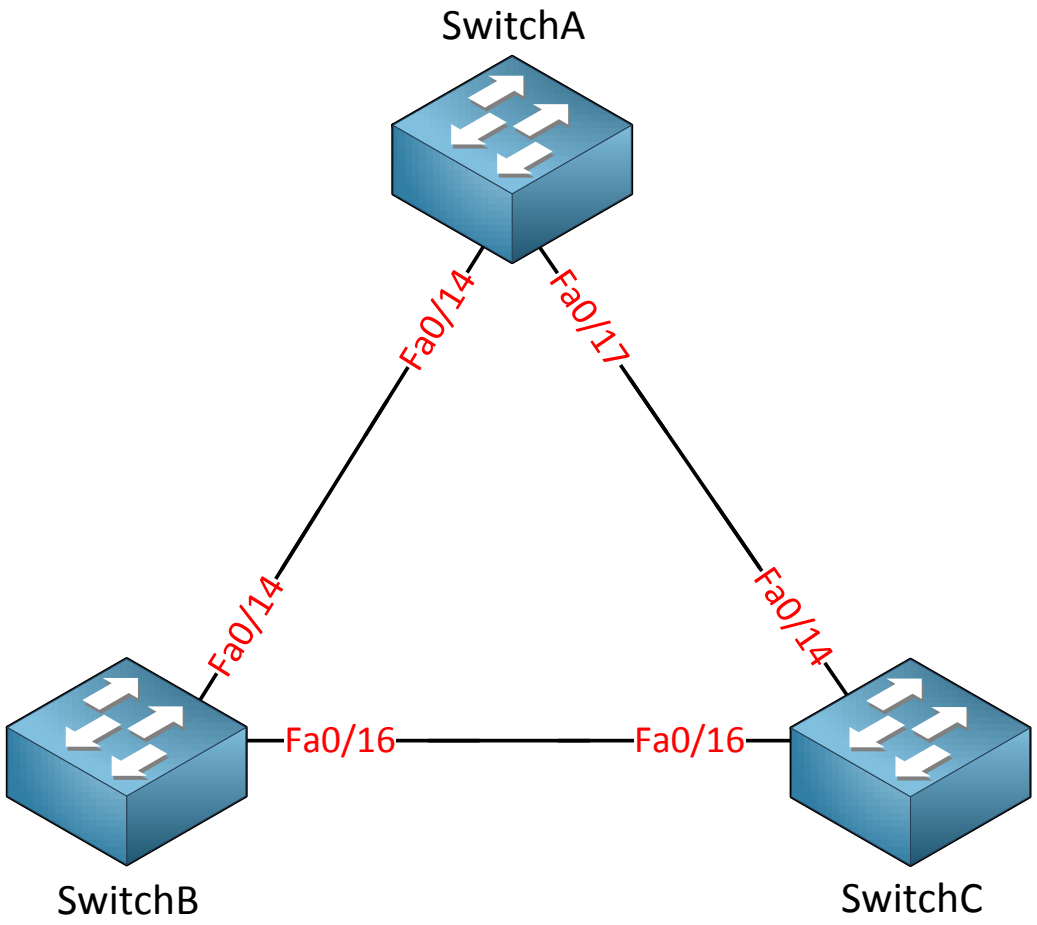
When an interface is in blocking mode and the topology changes, it's possible that an interface that is currently in blocking mode has to move to the forwarding state. When this is the case, the blocking mode will last for 20 seconds before it moves to the listening state. This means that it takes 20 (blocking) + 15 (listening) + 15 (learning) = 50 seconds before the interface is in the forwarding state.

Any modern PC boots much faster than 30 seconds. Here's an overview of the port states:

State	Forward Frames	Learn MAC addresses	Duration
Blocking	No	No	20 seconds
Listening	No	No	15 seconds
Learning	No	Yes	15 seconds
Forwarding	Yes	Yes	-

There is a way to speed up this process; I'll show you how to do this later.

What do you think of spanning-tree so far? In the next part of this chapter I'm going to look at some real switches and walk you through the configuration.



This is the topology we will use. Spanning-tree is enabled by default; let's start by checking some show commands.

```
SwitchA#show spanning-tree

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    32769
              Address     000f.34ca.1000
              Cost        19
              Port        19 (FastEthernet0/17)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32769  (priority 32768 sys-id-ext 1)
              Address     0011.bb0b.3600
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
---
Fa0/14              Desg FWD 19        128.16   P2p
Fa0/17              Root FWD 19        128.19   P2p
```

The **show spanning-tree** command is the most important show command to remember. There's quite some stuff here so I'm going to break it down for you!

```
VLAN0001
  Spanning tree enabled protocol ieee
```

We are looking at the spanning-tree information for VLAN 1. Spanning-tree has multiple versions and the default version on Cisco switches is PVST (Per VLAN spanning-tree). This is the spanning-tree for VLAN 1.

```
Root ID      Priority    32769
             Address     000f.34ca.1000
             Cost        19
             Port        19 (FastEthernet0/17)
```

Here you see the **information of the root bridge**. You can see that it has a priority of 32769 and its MAC address is 000f.34ca.1000. From the perspective of SwitchA it has a cost of 19 to reach the root bridge. The port that leads to the root bridge is called the root port and for SwitchA this is fa0/17.

```
  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     0011.bb0b.3600
```

This part shows us the **information about the local switch**, SwitchA in our case. There's something funny about the priority here….you can see it show two things:

- Priority 32769
- Priority 32768 sys-id-ext 1

The **sys-id-ext** value that you see is the VLAN number. The priority is 32768 but spanning-tree will add the VLAN number so we end up with priority value 32769. Last but not least we can see the MAC address of SwitchA which is 0011.bb0b.3600.

```
Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
```

Here's some information on the different times that spanning-tree uses:

- **Hello time:** every 2 seconds a BPDU is sent.
- **Max Age:** If we don't receive BPDUs for 20 seconds we know something has changed in the network and we need to re-check the topology.
- **Forward Delay:** This is the time spent in the "listening" and "learning" states. By default it's 15 seconds.

```
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
---
Fa0/14              Desg FWD 19        128.16   P2p
Fa0/17              Root FWD 19        128.19   P2p
```

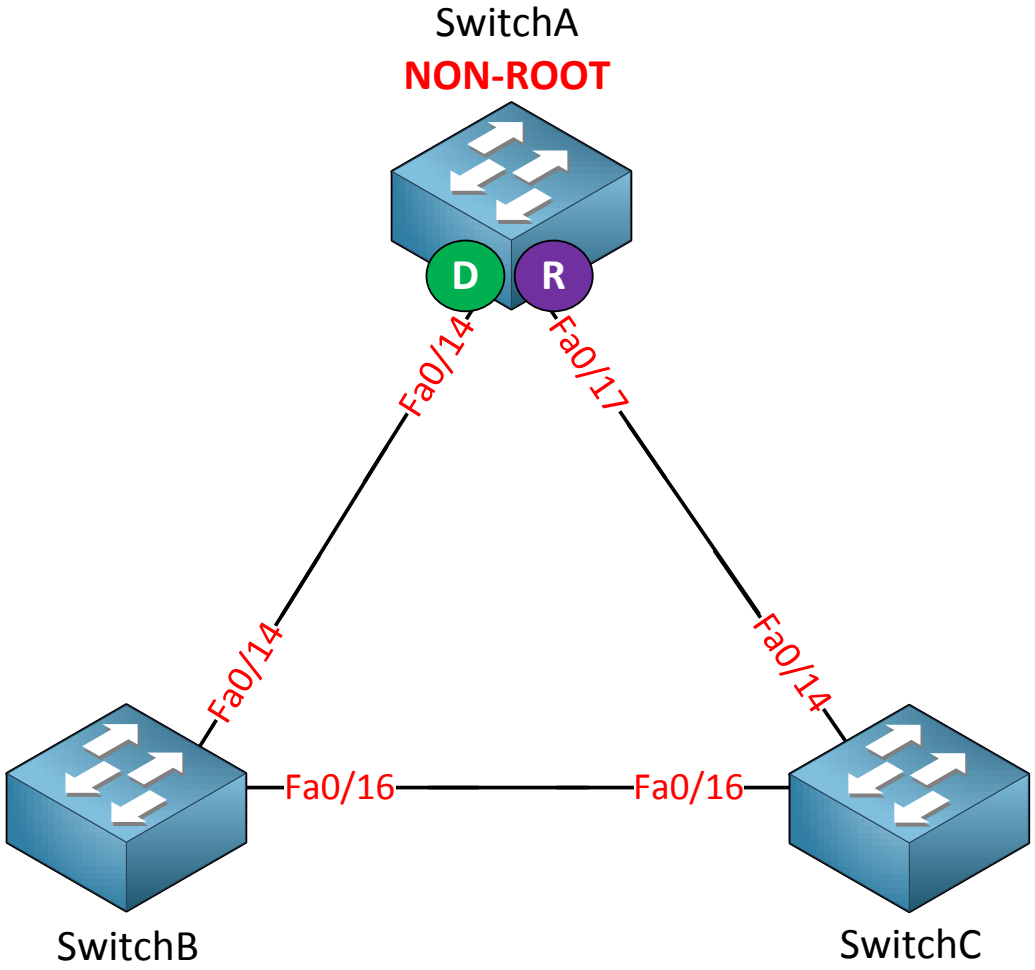
The last part of the show spanning-tree commands shows us the interfaces and their status.

SwitchA has two interfaces:

- Fa0/14 is a **designated** port and in **(FWD) forwarding mode**.
- Fa0/17 is a **root** port and in **(FWD) forwarding mode.**

The **prio.nbr** you see here is the **port priority** that I explained earlier. We'll play with this in a bit.

Because only non-root switches have a root-port I can conclude that SwitchA is a non-root switch. I know that fa0/17 on SwitchA leads to the root bridge.



For the sake of having a good overview I just added what we saw in the show spanning-tree command in the picture above. We know that SwitchA is a non-root, fa0/14 is a designated port and fa0/17 is a root port.

```
SwitchB#show spanning-tree

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    32769
             Address     000f.34ca.1000
             Cost        19
             Port        18 (FastEthernet0/16)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             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
------------------- ---- --- --------- -------- --------------------------------
---
Fa0/14              Altn BLK 19        128.16   P2p
Fa0/16              Root FWD 19        128.18   P2p
```

Let's take a look at SwitchB…what do we have here?

```
  Root ID    Priority    32769
             Address     000f.34ca.1000
             Cost        19
             Port        18 (FastEthernet0/16)
```

Here we see information about the root bridge. This information is similar to what we saw on SwitchA. The root port for SwitchB seems to be fa0/16.

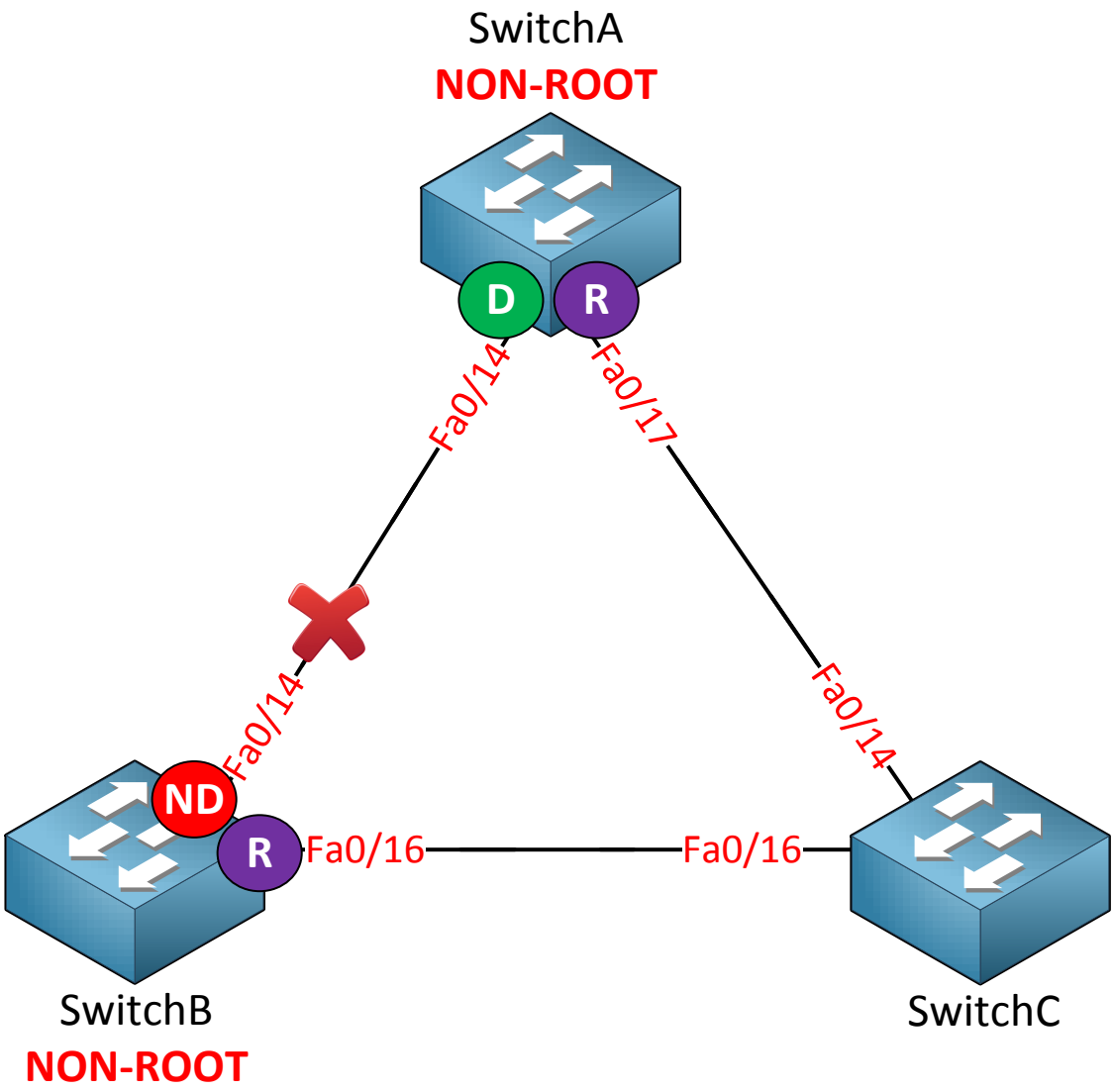
```
  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     0019.569d.5700
```

This is the information about SwitchB. The priority is the same as on SwitchA, only the MAC address (0019.569d.5700) is different.

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

This part looks interesting; there are two things we see here:

- Interface fa0/14 is an **non-designated** port and in **(BLK) blocking** mode. Cisco IOS switches will show the role as ALTN (Alternate port) but in reality this is a non-designated port. We'll talk about the alternate port later when we discuss rapid spanning-tree.
- Interface fa0/16 is a **root** port and in **(FWD) forwarding** mode.



With the information we just found on SwitchB we can add more items to our topology picture. We are almost finished!

```
SwitchC#show spanning-tree

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    32769
              Address     000f.34ca.1000
              This bridge is the root
              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 300

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Fa0/14           Desg FWD 19        128.14   P2p
Fa0/16           Desg FWD 19        128.16   P2p
```

Let's break down what we have here:

```
  Root ID     Priority    32769
              Address     000f.34ca.1000
              This bridge is the root
```

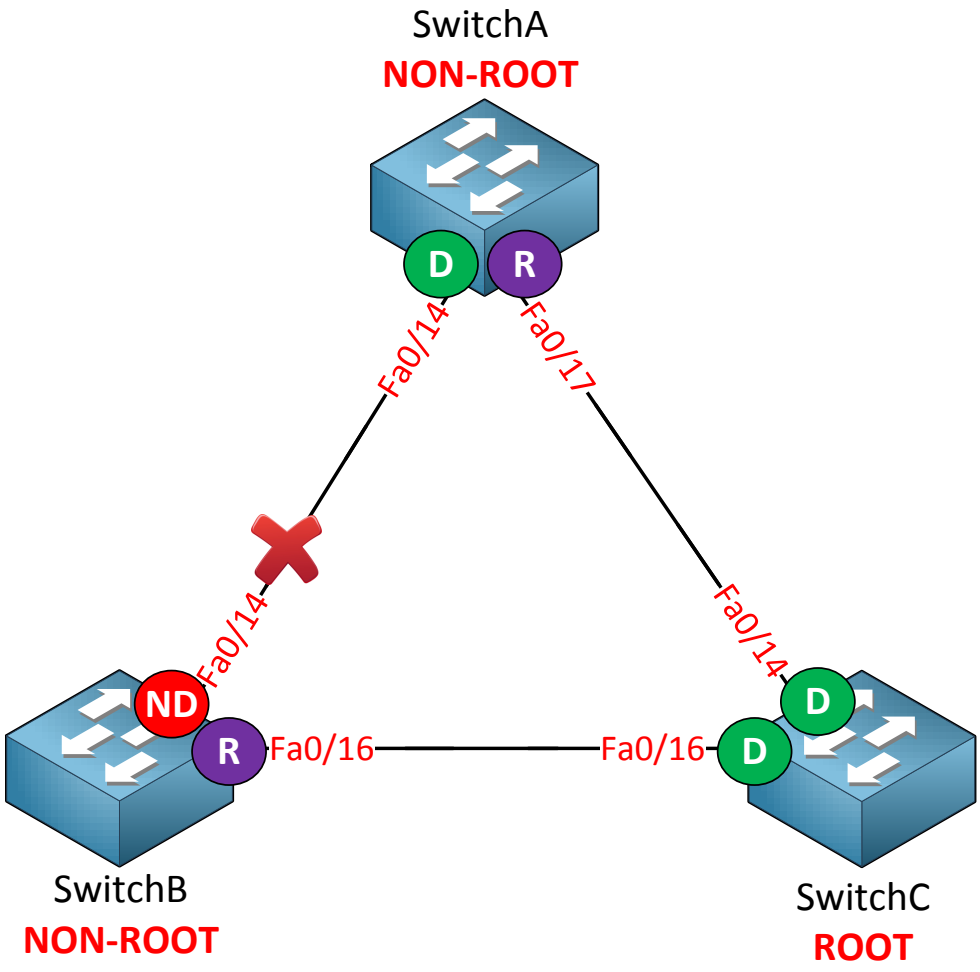
Bingo…SwitchC is the root bridge in this network. We already knew that because SwitchA and SwitchB are both non-root but this is how we verify it by looking at SwitchC.

```
Bridge ID     Priority    32769  (priority 32768 sys-id-ext 1)
              Address     000f.34ca.1000
```

We can also see the MAC address of SwitchC.

```
Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Fa0/14           Desg FWD 19        128.14   P2p
Fa0/16           Desg FWD 19        128.16   P2p
```

Both interfaces on SwitchC are **designated ports** and in **(FWD) forwarding** mode.



Our picture is now complete. We successfully found out what the spanning-tree topology looks like by using the show spanning-tree command! Why was SwitchC chosen as the root bridge? We have to look at the bridge identifier for the answer:

```
SwitchA#show spanning-tree | begin Bridge ID
  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     0011.bb0b.3600
```

```
SwitchB#show spanning-tree | begin Bridge ID
  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     0019.569d.5700
```

```
SwitchC#show spanning-tree | begin Bridge ID
  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     000f.34ca.1000
```

The priority is the same on all switches (32768) so we have to look at the MAC addresses:

- SwitchA: 0011.bb0b.3600
- SwitchB: 0019.569d.5700
- SwitchC: 000f.34ca.1000

SwitchC has the lowest MAC address so that's why it became root bridge. Why was the fa0/14 interface on SwitchB blocked and not the fa0/14 interface on SwitchA? Once again we have to look at the bridge identifier. The priority is 32768 on both switches so we have to compare the MAC address:

- SwitchA: 0011.bb0b.3600
- SwitchB: 0019.569d.5700

SwitchA has a lower MAC address and thus a better bridge identifier. That's why SwitchB lost this battle and has to shut down its fa0/14 interface.

What if I want another switch to become root bridge? For example SwitchA:

```
SwitchA(config)#spanning-tree vlan 1 root primary
```

There are two methods so I can change the root bridge. The **spanning-tree vlan root primary** command is the first one. This is a **macro that runs only once and** that looks at the current priority of the root bridge and changes your running-config to lower your own priority. Because we use PVST (Per VLAN Spanning-Tree) we can change this for each VLAN.

```
SwitchA#show spanning-tree

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    24577
              Address     0011.bb0b.3600
              This bridge is the root
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    24577  (priority 24576 sys-id-ext 1)
```

You can see that SwitchA is now the root bridge because its priority has been changed to 24576. It has decreased its priority by 4096 to become the root bridge.

```
SwitchA#show run | include priority
spanning-tree vlan 1 priority 24576
```

If you look at the running-config you can see that that the spanning-tree vlan root primary command/macro changed the priority for us.

```
SwitchA(config)#spanning-tree vlan 1 priority ?
  <0-61440>  bridge priority in increments of 4096

SwitchA(config)#spanning-tree vlan 1 priority 4096
```

Changing the priority manually is the second method.

Just type in the **spanning-tree vlan priority** command and set it to whatever value you like.

```
SwitchA#show spanning-tree

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    4097
              Address     0011.bb0b.3600
              This bridge is the root
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    4097   (priority 4096 sys-id-ext 1)
              Address     0011.bb0b.3600
```

We can verify this by checking the show spanning-tree command once again.

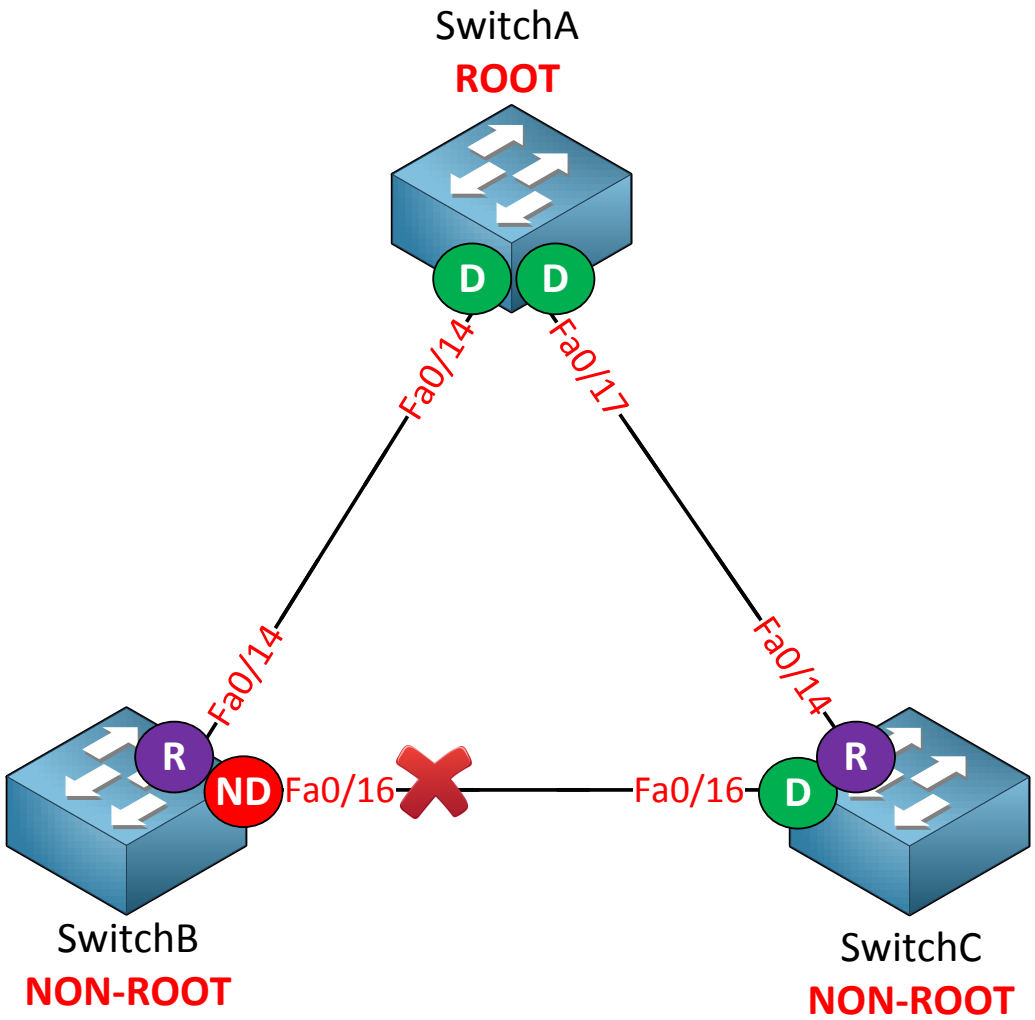
Because SwitchA is now the root bridge our spanning-tree topology looks different:

```
SwitchA#show spanning-tree | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
---
Fa0/14              Desg FWD 19        128.16   P2p
Fa0/17              Desg FWD 19        128.19   P2p
```

```
SwitchB#show spanning-tree | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
---
Fa0/14              Root FWD 19        128.16   P2p
Fa0/16              Altn BLK 19        128.18   P2p
```

```
SwitchC#show spanning-tree | begin Interface
Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Fa0/14           Root FWD 19        128.14   P2p
Fa0/16           Desg FWD 19        128.16   P2p
```

This is all the information we need. Let's update our topology picture…



Let's play some more with spanning-tree! What if I want to change the root port on SwitchB so it reaches the root bridge through SwitchC? From SwitchB's perspective it can reach the root bridge through fa0/14 (cost 19) or by going through fa0/16 (cost 19+19 = 38). Let's change the cost and see what happens.

```
SwitchB(config)#interface fa0/14
SwitchB(config-if)#spanning-tree cost 500
```

Let's change the cost of the fa0/14 interface by using the **spanning-tree cost** command.

```
SwitchB#show spanning-tree | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
---
Fa0/14              Altn BLK 500       128.16   P2p
Fa0/16              Root FWD 19        128.18   P2p
```

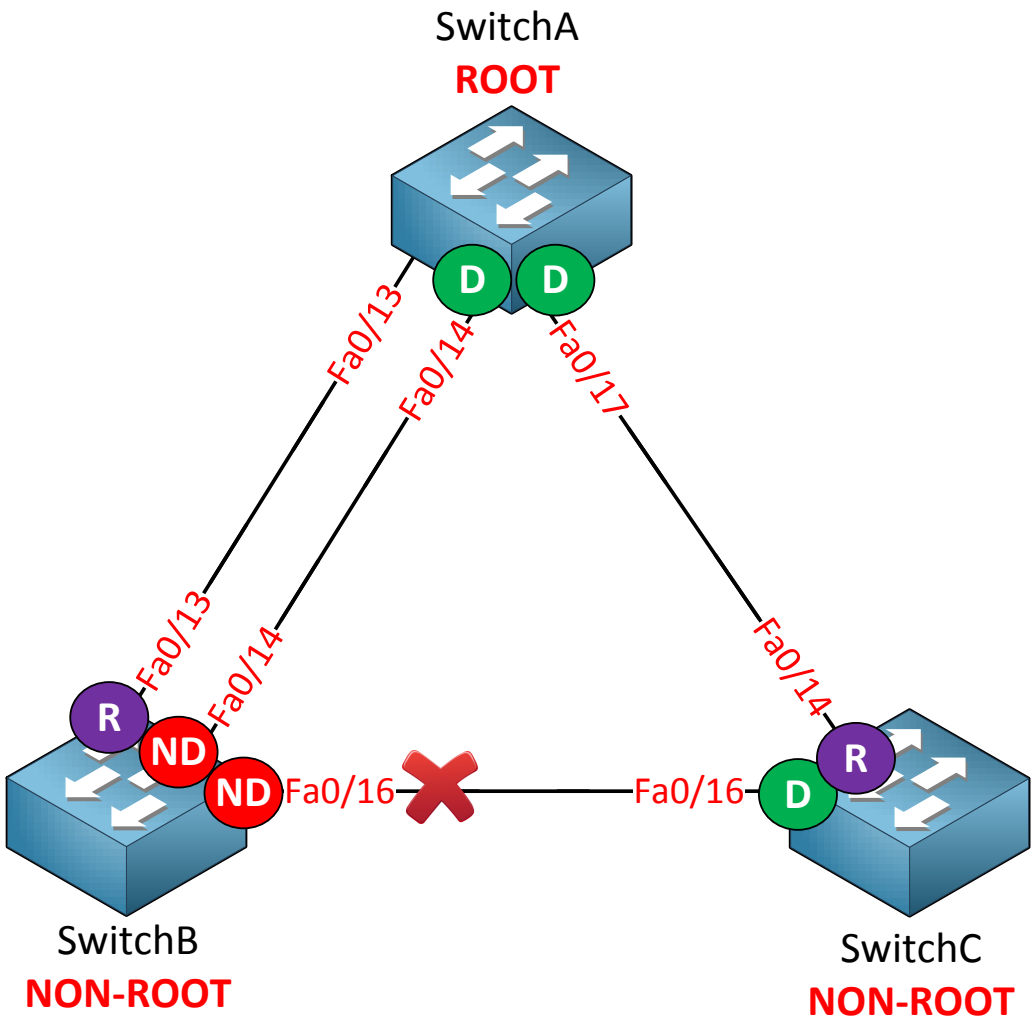
You can see that the fa0/14 now has a cost of 500 and it has been blocked. Fa0/16 is now the root port.

```
SwitchB#show spanning-tree

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    4097
              Address     0011.bb0b.3600
              Cost        38
```

To reach the root the total cost is now 38.

```
SwitchB(config)#interface fa0/14
SwitchB(config-if)#no spanning-tree cost 500
```

Let's get rid of the higher cost before we continue.



I have added another cable between SwitchA and SwitchB. In the picture above you can see that fa0/13 is now the root port. Fa0/14 has been blocked.

```
SwitchB#show spanning-tree | begin Interface
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
```

Why did fa0/13 become the root port instead of fa0/14? The cost to reach the root bridge is the same on both interfaces. The answer lies in the port priority:

- Fa0/13: port priority 128.15
- Fa0/14: port priority 128.16

The "128" is a default value which we can change. 15 and 16 are the port numbers, each interface is assigned a port number. Fa0/13 has a lower port priority so that's why it was chosen.

Let's change the port priority and see what happens:

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#spanning-tree port-priority 16
```

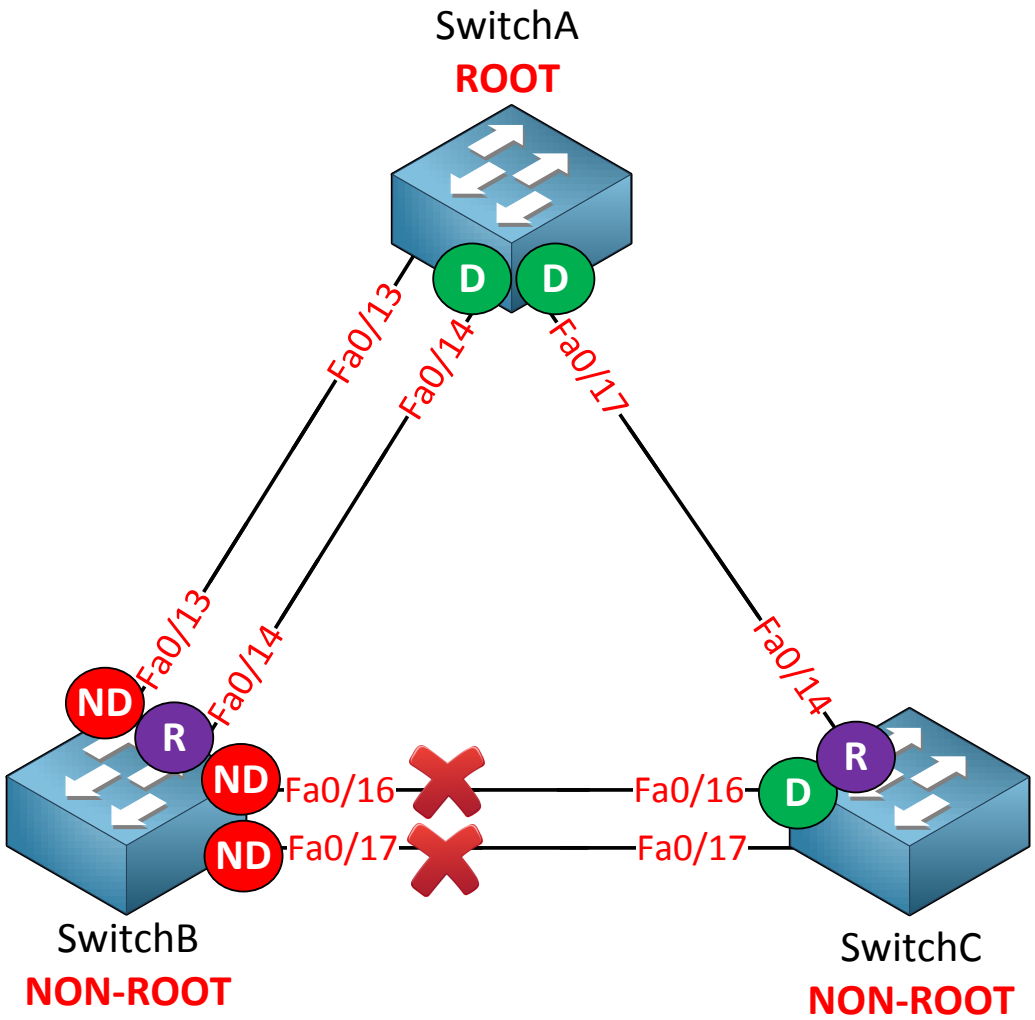
Note that I'm changing the port priority on SwitchA, not on SwitchB. At the moment SwitchB is receiving a BPDU on its fa0/13 and fa0/14 interfaces. Both BPDUs are the same. By changing the port priority on SwitchA, SwitchB will receive a BPDU with a better port priority on its fa0/14 interface.

```
SwitchA#show spanning-tree | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
---
Fa0/13              Desg FWD 19        128.15   P2p
Fa0/14              Desg FWD 19         16.16   P2p
```

You can see the port priority has been changed on SwitchA.

```
SwitchB#show spanning-tree | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
---
Fa0/13              Altn BLK 19        128.15   P2p
Fa0/14              Root FWD 19        128.16   P2p
Fa0/16              Altn BLK 19        128.18   P2p
```

Interface fa0/14 on SwitchB is now the root port!

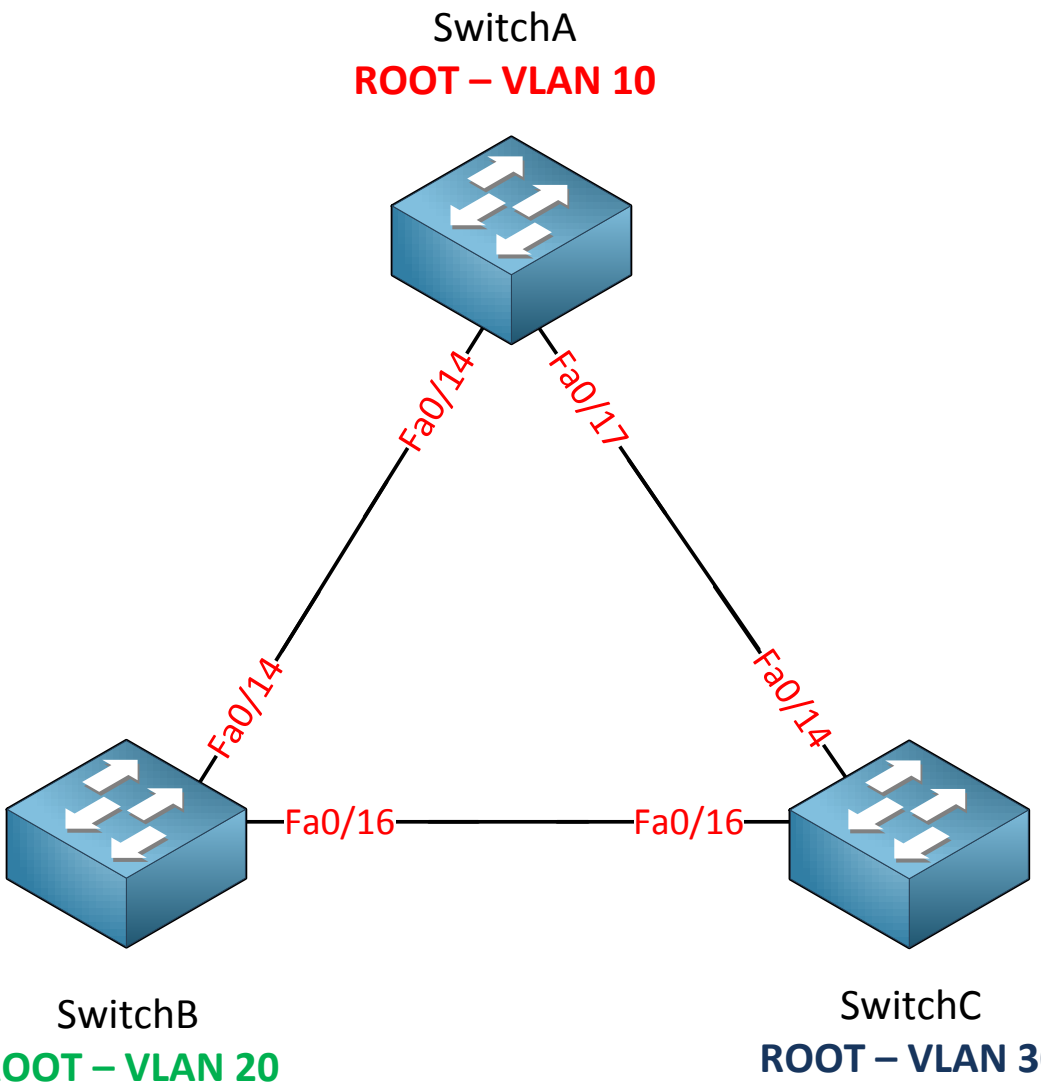


In the picture above I added another cable between SwitchB and SwitchC. This interface will also become a non-designated port and it will be blocked.

```
SwitchB#show spanning-tree | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
---
Fa0/13              Altn BLK 19        128.15   P2p
Fa0/14              Root FWD 19        128.16   P2p
Fa0/16              Altn BLK 19        128.18   P2p
Fa0/17              Altn BLK 19        128.19   P2p
```

We can verify our configuration here. Just another blocked port…

Are you following me so far? I hope so! If you are having trouble understanding the different spanning-tree commands I recommend you to build the same topology as the one I'm using above and to take a look at your own spanning-tree topology. Play with the priority, cost and port priority to see what the result will be.



Let's get back to the basics. I have resetted all switches back to factory default settings because I want to show you how spanning-tree works with multiple VLANs. In the previous example we were only using VLAN 1. Now I'm going to add VLAN 10, 20 and 30 and each switch will become root bridge for a VLAN.

```
SwitchA(config)#vlan 10
SwitchA(config-vlan)#vlan 20
SwitchA(config-vlan)#vlan 30
```

```
SwitchB(config)#vlan 10
SwitchB(config-vlan)#vlan 20
SwitchB(config-vlan)#vlan 30
```

```
SwitchC(config)#vlan 10
SwitchC(config-vlan)#vlan 20
SwitchC(config-vlan)#vlan 30
```

First I'm going to create all VLANs. If you are running VTP server/client mode you only have to do this on one switch.

```
SwitchA(config)#interface fa0/14
SwitchA(config-if)#switchport trunk encapsulation dot1q
SwitchA(config-if)#switchport mode trunk
SwitchA(config)#interface fa0/17
SwitchA(config-if)#switchport trunk encapsulation dot1q
SwitchA(config-if)#switchport mode trunk
```

```
SwitchB(config)#interface fa0/14
SwitchB(config-if)#switchport trunk encapsulation dot1q
SwitchB(config-if)#switchport mode trunk
SwitchB(config)#interface fa0/16
SwitchB(config-if)#switchport trunk encapsulation dot1q
SwitchB(config-if)#switchport mode trunk
```

```
SwitchC(config)#interface fa0/14
SwitchC(config-if)#switchport trunk encapsulation dot1q
SwitchC(config-if)#switchport mode trunk
SwitchC(config)#interface fa0/16
SwitchC(config-if)#switchport trunk encapsulation dot1q
SwitchC(config-if)#switchport mode trunk
```

Make sure the interfaces between the switches are trunks. Mine were access interfaces so I changed them to trunk mode myself.

```
SwitchA#show spanning-tree summary | begin Name
Name                   Blocking Listening Learning Forwarding STP Active
---------------------- -------- --------- -------- ---------- ----------
VLAN0001                    0         0        0          2          2
VLAN0010                    0         0        0          2          2
VLAN0020                    0         0        0          2          2
VLAN0030                    0         0        0          2          2
---------------------- -------- --------- -------- ---------- ----------
4 vlans                     0         0        0          8          8
```

```
SwitchB#show spanning-tree summary | begin Name
Name                   Blocking Listening Learning Forwarding STP Active
---------------------- -------- --------- -------- ---------- ----------
VLAN0001                    1         0        0          1          2
VLAN0010                    1         0        0          1          2
VLAN0020                    1         0        0          1          2
VLAN0030                    1         0        0          1          2
---------------------- -------- --------- -------- ---------- ----------
4 vlans                     4         0        0          4          8
```

```
SwitchC#show spanning-tree summary | begin Name
Name                   Blocking Listening Learning Forwarding STP Active
---------------------- -------- --------- -------- ---------- ----------
VLAN0001                    0         0        0          2          2
VLAN0010                    0         0        0          2          2
VLAN0020                    0         0        0          2          2
VLAN0030                    0         0        0          2          2
---------------------- -------- --------- -------- ---------- ----------
4 vlans                     0         0        0          8          8
```

You can use the **show spanning-tree summary** command for a quick overview of the spanning-tree topologies. You can also just use the show spanning-tree command and you will get information on all the VLANs. As you can see my switches have created a spanning-tree topology for each VLAN.

```
SwitchC#show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol ieee
  Root ID    Priority    32778
             Address     000f.34ca.1000
             This bridge is the root
```

```
SwitchC#show spanning-tree vlan 20

VLAN0020
  Spanning tree enabled protocol ieee
  Root ID    Priority    32788
             Address     000f.34ca.1000
             This bridge is the root
```

```
SwitchC#show spanning-tree vlan 30

VLAN0030
  Spanning tree enabled protocol ieee
  Root ID    Priority    32798
             Address     000f.34ca.1000
             This bridge is the root
```

Some show commands reveal to us that SwitchC is the root bridge for VLAN 10, 20 and 30.

```
SwitchA(config)#spanning-tree vlan 10 priority 4096
```

Let's lower the priority on SwitchA for VLAN 10 to 4096 so it will become the root bridge.

```
SwitchA#show spanning-tree vlan 10 | include root
             This bridge is the root
```

Here's a quick way to verify our configuration.

```
SwitchB(config)#spanning-tree vlan 20 priority 4096
```

```
SwitchB#show spanning-tree vlan 20 | include root
             This bridge is the root
```

SwitchB is the root for VLAN 20.

```
SwitchC(config)#spanning-tree vlan 30 priority 4096
SwitchC#show spanning-tree vlan 30 | include root
             This bridge is the root
```

And last but not least here is SwitchC as the root bridge for VLAN 30.

That's all there is to it! Of course different interfaces will be blocked because we have a different root bridge for each VLAN. I'm not going to try to create a picture that shows all the designated/non-designated/root ports for all VLANs because we'll end up with a Picasso-style picture!

We can change the configuration of our spanning-tree configuration per VLAN. For example I can tune the timers if I want to speed up the spanning-tree process:

```
SwitchA#show spanning-tree vlan 10 | begin Root ID
  Root ID     Priority    4106
              Address     0011.bb0b.3600
              This bridge is the root
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
```

Let's change these default timers.

```
SwitchA(config)#spanning-tree vlan 10 hello-time 1
```

The hello time specifies how often a BPDU is sent. The default is 2 seconds but I changed it to 1 second.

```
SwitchA#show spanning-tree vlan 10 | begin Root ID
  Root ID     Priority    4106
              Address     0011.bb0b.3600
              This bridge is the root
              Hello Time   1 sec  Max Age 20 sec  Forward Delay 15 sec
```

Our configuration is successful! These changes are only applied to VLAN 10.

```
SwitchB(config)#spanning-tree vlan 20 max-age 6
```

We can also change the max-age timer. When a switch no longer receives periodic BPDUs on a switch it will wait for the max-age timer before it decides to re-check the spanning-tree topology. The default is 20 seconds but we can change it to 6 seconds.

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

Click on the link below to get the full version.

Get How to Master CCNP SWITCH Today