ETHERCHANNEL: LACP, PAGP, AND STATIC PROTOCOLS

BY ANDREW STIBBARDS, SUNSET LEARNING INSTITUTE CISCO SPECIALIZED INSTRUCTOR

This article will be broken into two parts. Part 1 defines what EtherChannel is and cases for using it in your networks from a non-technical perspective. Part 2 delves into the actual details of the protocols and how to configure them on your devices. So if you are already familiar with the protocols general background and application head on over to Part 2. If not then please start with Part 1 to understand why you should know about this tool.

PART 1

One of the general ideas within networks is that we always want to provide fast and uninterrupted service to our users, without going over budget. So there is a question, "How do we do that?" One of the solutions to that question is to add more devices with more uplinks to your network, but that can only scale to a certain level. Even exchanging older devices for newer with faster interfaces, such as moving from FastEthernet interfaces to 10 Gigabit interfaces, only exists as a possible solution until you actually implement it, then you can no longer keep trading up. So another solution is to start increasing the number of physical cables between your network devices, because obviously more paths equals more options which means faster traffic, right? Wrong. Protocols such as Spanning-Tree Protocol (STP) will logically block extra interfaces in order to avoid loops on the network, and our dynamic routing protocols (OSPF, EIGRP, and RIP) only use the best path when multiple exist. The routing protocols do have the ability to load-balance traffic when multiple paths exist, but in that case the two paths have to have EXACTLY the same metric or value from the routing protocols perspective, or have metrics that are very close to each other. This is sometimes hard to find, depending on your networks physical layout. So this is the question: how to achieve the goals of faster, reliable communication without going over budget, and without breaking any protocols that we are already using in our networks. The solution is EtherChannel.

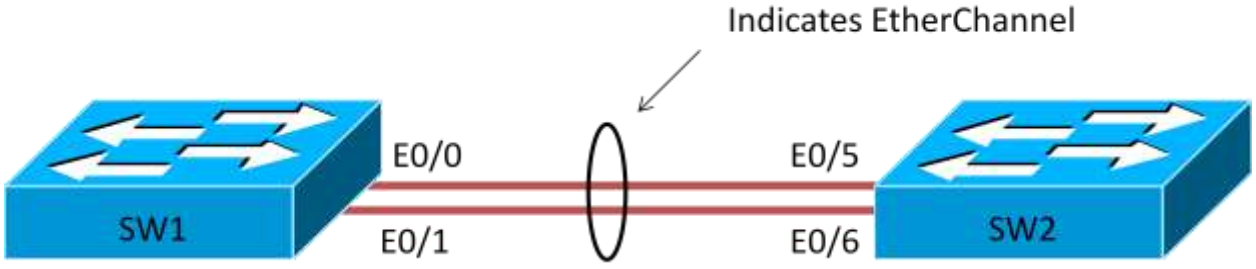


Figure 1: Sample EtherChannel Physical Layout

The idea of Etherchannel is simple in its design, yet powerful in effect. The short definition of EtherChannel is multiple physical switch interfaces, bundled together, behaving for all intents and purposes as one logical interface. So what is necessary to pull this off? Simply, you run more connections between your existing physical devices, and then configure those interfaces to appear as a single interface. So you don't have to buy more equipment, just the cabling. In the world of STP, this new interface looks like only one interface. Now in the real world all of the physical ports are active, forwarding traffic, but due to the algorithms and device behavior it doesn't introduce loops, when correctly configured. From the routing protocols perspective, it looks like one high speed path between the devices, so the routing protocol would choose to use it. In the real world, when the routing protocol sends traffic over the bundle, the traffic is spread across all the physical ports. And here is maybe the best part. In the world of redundancy, if at least one of the physical interfaces is still active, the overall Etherchannel appears to be active. So at that 4 a.m. moment (nothing good happens at 4 a.m.) when a cable or interface stops working, the overall path between your switches is still there!

So what is required to pull this off? All you need in your current network is to have some extra interfaces on your switches, and a cable to run between them. The configuration is simple, and you immediately have that speed and redundancy you wanted, without breaking the budget. Next up we are going to look at the configuration and some background technical information of different EtherChannel types.

Part 2

There are 3 methods to configure EtherChannel in your network: the PAgP and LACP negotiating protocols, or Static mode. The first step is assigning the physical interfaces to the Etherchannel, with different keywords indicating which protocol is to be used. Let's examine the 3 methods and their differences.

Static

A sample static EtherChannel configuration for Figure 1 would be:

```
SW1(config)#interface range e0/0-1

SW1(config-if-range)#channel-group 1 mode on

SW1(config-if-range)#interface port-channel 1

SW1(config-if)#switchport mode trunk
```

You would then do the same config on SW2, just with the different port numbers. So what did that do? First you grabbed the physical interfaces with the range command, then added them to the

Etherchannel using the channel-group command. The 1 referenced which Etherchannel they were joining, and the mode of **on** indicates static Etherchannel. At that point you went to the logical interface for managing the bundle, port-channel 1, and made it a trunk. All configuration applied to the logical port-channel interface is automatically applied to the physical interfaces participating in the bundle.

Now, there is nothing "wrong" with this setup. However, because there is no negotiating protocol, there are two possible problems. One, if the physical interfaces bundled have different configurations, the device will still bundle them. This would be a problem if we statically bundled an access port and a trunk port, or an interface operating at 10 Mbps and one operating at 100 Mbps. Two, due to normal switch behavior, if the correct corresponding interfaces on the other side are not done correctly, you can introduce a switching loop. So we will list this method as "possible, but not recommended."

PAgP

The Port-Aggregation Protocol (PAgP) is Cisco proprietary. It uses the multicast address of

01-00-0C-CC-CC-CC. A sample config for using PAgP for Figure 1 would be:

SW1(config)#interface range e0/0-1

SW1(config-if-range)#channel-group 1 mode (desirable/auto)

SW1(config-if-range)#interface port-channel 1

SW1(config-if)#switchport mode trunk

Almost identical in the commands, the only difference was to use either the keyword **desirable** or **auto** instead of **on**. The **desirable** keyword indicates using the PAgP protocol, where the interface is sending requests to the other side to see if it is also using PAgP. The **auto** keyword defines using PAgP, but the interface is not sending requests. So if you have **auto** on both sides, you will never get an Etherchannel. You need at least one side using **desirable**, or both.

The main reasons we love this protocol over Static configuration is that is one, it does a configuration check on participating interfaces, as well as confirms that the neighboring interfaces are also using PAgP. That means that it guarantees that interfaces that don't have similar configurations will not participate, and we won't get an accidental switching loop.

The downside to PAgP is that it does not support participating interfaces being on different physical switches. So if you are using VSS, or Stackwise, or any setup where you have device redundancy through virtualization, then it won't work.

LACP

Which brings us to the Link Aggregation Control Protocol (LACP). LACP is an open protocol, published under the 802.3ad specification. It uses the multicast address of 01-80-c2-00-00-02. The sample config for LACP for Figure 1 would be:

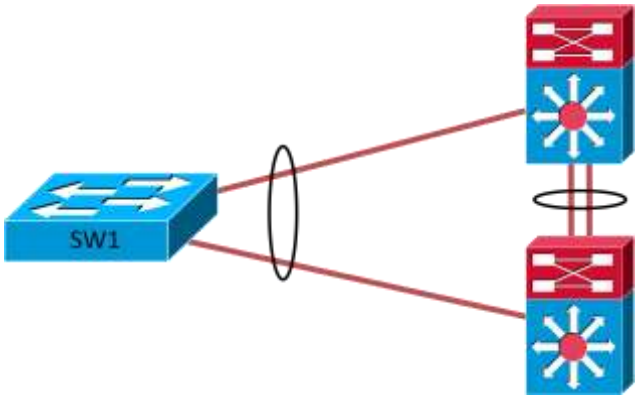
SW1(config)#interface range e0/0-1

SW1(config-if-range)#channel-group 1 mode (active/passive)

SW1(config-if-range)#interface port-channel 1

SW1(config-if)#switchport mode trunk

As you can see, again the configuration is almost exactly the same. The difference this time is the keywords of **active** and **passive.** The **active** keyword indicates the use of the LACP protocol on the interface, and the interface is negotiating with the other side, trying to form an Etherchannel. The **passive** keyword indicates using LACP, but only responding to requests, not sending any. Sounds almost exactly PAgP, right? Except for one major difference: LACP supports having participating ports on multiple physical switches. An example of that would be:



In this situation SW1 has only 1 EtherChannel, but in reality it is running to two different distribution switches. With this model you can achieve both line and device redundancy. Since PAgP doesn't support this, we will stick to using LACP. Static EtherChannel does support this type of physical setup, but it still has the two problems it had before (possible misconfigured interfaces participating, possible switching loops), so we will avoid using it if possible.

The last note I want to make now is about the physical and logical interface configuration. In order for a physical interface to be bundled into the Etherchannel, it must have EXACTLY THE SAME commands as the port-channel logical interface. If one command is different, it will not participate. This is why we like LACP and PAgP, because they are aware of dissimilar configurations. Static EtherChannel does not. Simply, you can do a **#show run** command, and read each line on the interfaces. If they are the same, you should have no problems.

Summary

So at this point you know how to make an operational EtherChannel. Identify the existing open interfaces on your switches. Run another line between the devices. Assign the physical ports to the EtherChannel, using the appropriate keyword for the method you want to use. Once it is operational STP and your routing protocols will do what they do best, and automatically calculate and re-converge for this faster "single" link that just came online. At that point you have added redundancy and speed to your network without adding a single device.

Now operational EtherChannel does not mean optimized. My next article will explore the default method of load-balancing that EtherChannel uses, and how to modify that for better throughput.