- EtherChannel -

Port Aggregation

A network will often span across multiple switches. Trunk ports are usually used to connect switches together.

There are two issues with using only a *single* physical port for the trunk connection:

- The port represents a **single point of failure.** If the port goes down, the trunk connection is lost.
- The port represents a traffic **bottleneck**. All other ports on the switch will use that one port to communicate across the trunk connection.

Thus, the obvious benefits of adding redundancy to the trunk connection are **fault tolerance** and **increased bandwidth**, via load balancing.

However, simply trunking two or more ports between the switches will not work, as this creates a **switching loop**. One of two things will occur:

- Spanning Tree Protocol (STP) will disable one or more ports to eliminate the loop.
- If STP is disabled, the switching loop will result in an almost instantaneous broadcast storm, crippling the network.

Port aggregation allows multiple *physical* ports to be bundled together to form a single *logical* port. The switch and STP will treat the bundled ports as a single interface, eliminating the possibility of a switching loop.

Cisco's implementation of port aggregation is called **EtherChannel.** EtherChannel supports **Fast**, **Gigabit**, and **10 Gigabit** Ethernet ports.

A maximum of **8 active ports** are supported in a single EtherChannel. If the ports are operating in full duplex, the maximum *theoretical* bandwidth supported is as follows:

- Fast Ethernet – **1600 Mbps**
- Gigabit Ethernet – **16 Gbps**
- 10 Gigabit Ethernet – **160 Gbps**

The maximum number of supported EtherChannels on a single switch is platform-dependent, though most support up to **64** or **128** EtherChannels.

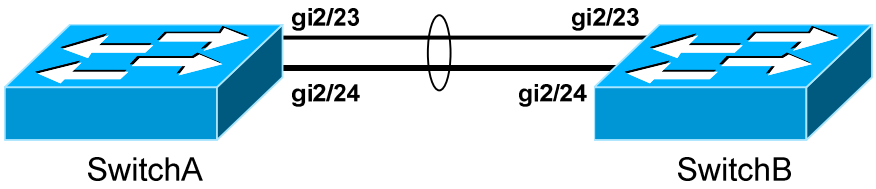
* * *

EtherChannel Requirements

The previous section described the benefits of port aggregation for a trunk connection. However, EtherChannels can be formed with *either* access or trunk ports. EtherChannels are also supported on Layer-3 interfaces.

Implementing an EtherChannel for access ports provides increased bandwidth and redundancy to a host device, such as a server. However, the host device must support a **port aggregation protocol**, such as LACP. Port aggregation protocols are covered in great detail later in this guide.

Similarly, implementing EtherChannel for trunk connections provides increased bandwidth and redundancy to other switches.



If a port in an EtherChannel bundle fails, traffic will be redistributed across the remaining ports in the bundle. This happens nearly *instantaneously*.

For an EtherChannel to become active, all ports in the bundle **must be configured identically**, regardless if the EtherChannel is being used with access or trunk ports. Port settings that must be identical include the following:

- **Speed** settings
- **Duplex** settings
- **STP** settings
- **VLAN membership** (for access ports)
- **Native VLAN** (for trunk ports)
- **Allowed VLANs** (for trunk ports)
- **Trunking encapsulation protocol** (for trunk ports)

On trunk connections, the above settings must be configured identically across all participating ports on *both* switches.

Historically, **port-security** has not been supported on an EtherChannel. Newer platforms may provide support as long as port-security is enabled on both the physical interfaces and the EtherChannel itself.

* * *

EtherChannel Load-Balancing

Traffic sent across an EtherChannel is *not* evenly distributed across all ports in the bundle. Instead, EtherChannel utilizes a load-balancing algorithm to determine the port to send the traffic out, based on one of several criteria:

- Source IP address - **src-ip**
- Destination IP address - **dst-ip**
- Source *and* destination IP address - **src-dst-ip**
- Source MAC address - **src-mac**
- Destination MAC address - **dst-mac**
- Source *and* Destination MAC address - **src-dst-mac**
- Source TCP/UDP port number - **src-port**
- Destination TCP/UDP port number - **dst-port**
- Source *and* destination port number - **src-dst-port**

Using a deterministic algorithm prevents perfect load-balancing. However, a particular traffic flow is forced to always use the same port in the bundle, preventing out-of-order delivery.

The default load-balancing method for a Layer-2 EtherChannel is either **src-mac** or **src-dst-mac**, depending on the platform. The default method for a Layer-3 EtherChannel is **src-dst-ip**.

The load-balancing method must be configured *globally* on the switch:

Switch(config)# *port-channel load-balance src-dst-mac*

To display the currently configured load-balancing method:

Switch# *show etherchannel load-balance*

```
EtherChannel Load-balancing Configuration:
       src-dst-mac
```

To view the load on each port in an EtherChannel (output abbreviated):

Switch# *show etherchannel 1 port-channel*

```
Index   Load   Port     EC state
------+------+--------+------------
  0     55     Gi2/23   active
  1     3A     Gi2/24   active
```

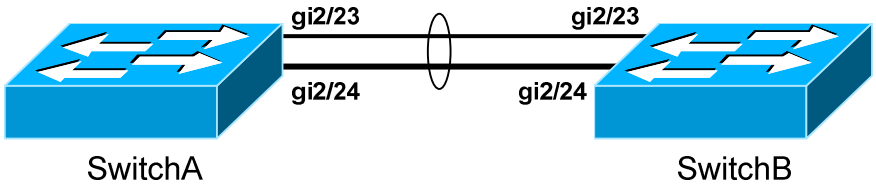
The load is rather cryptically represented in a hexadecimal value.

(Reference: http://www.cisco.com/c/en/us/support/docs/lan-switching/etherchannel/12023-4.html)

* * *

EtherChannel Load-Balancing Example

Consider the following example, where ports *gi2/23* and *gi2/24* on both switches are members of an EtherChannel:



Assume that the load-balancing method is **src-ip.** The first port in the EtherChannel will become Link0; the second will become Link1.

The two links in the EtherChannel can be represented in one binary **bit**. The load-balancing algorithm will create an index that associates Link0 with a binary bit of *0,* and Link1 with a bit of *1*.

As traffic passes through the EtherChannel, the algorithm will convert the source IP addresses into a binary hash, to compare against the index. For example, consider the following IP addresses and their binary equivalents:

10.1.1.2	00001010.00000001.00000001.00000010
10.1.1.3	00001010.00000001.00000001.00000011

Because there are only two ports in the EtherChannel, only one bit needs to be considered in the IP address – **the last bit.** The first address ends with a *0* bit, and thus would be sent out *Link0*. The second address ends with a *1* bit, and thus would be sent down *Link1*. Simple, right?

An EtherChannel that contained four ports would require a 2-bit index, requiring that the *last two* bits of the IP address be considered:

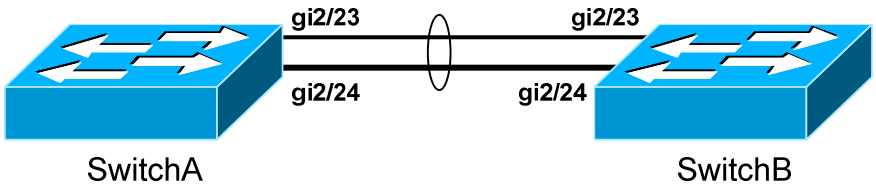
Link0	00
Link1	01
Link2	10
Link3	11

Best practice dictates that EtherChannels should contain an **even** number of ports, to promote uniform load-balancing. An EtherChannel can support an odd number of ports; however, this may result in one of the ports being severely overburdened due to uneven load-balancing.

* * *

EtherChannel Load-Balancing Example (continued)

Consider again the following example:



This time, assume that the load-balancing method is **src-dst-ip.** Again, the first port in the EtherChannel will become Link0; the second will become Link1.

Both the source *and* destination IP must be considered when choosing a link. This requires performing an **exclusive OR (XOR)** operation. In an XOR operation, if the two values being compared are *equal*, the result is *0.* If the two values are *not equal*, the result is *1*.

The following illustrates all possible results with a 1-bit index:

Source	0	1	0	1
Destination	0	0	1	1
Result	**0**	**1**	**1**	**0**

Consider the following source and destination IP address pair:

Source	192.168.1.10	11000000.10101000.00000001.00001010
Destination	192.168.2.25	11000000.10101000.00000010.00011001

The XOR result of the above address pair would be *1*. Thus, the traffic would be sent out *Link1*.

The following illustrates all possible results with a 2-bit index, representing four ports:

Source	00	00	00	00	01	01	01	01	10	10	10	10	11	11	11	11
Destination	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
Result	**00**	**01**	**10**	**11**	**01**	**00**	**11**	**10**	**10**	**11**	**00**	**01**	**11**	**10**	**01**	**00**

There are four possible results (*00, 01, 10, 11*), corresponding to the four ports in the EtherChannel.

Regardless of the load-balancing method used, **STP** will always send its packets through the **first operational port** in an EtherChannel bundle.

* * *

EtherChannel – Manual Configuration

There are two methods of configuring an EtherChannel:

- **Manually**
- **Dynamically**, using an aggregation protocol

To *manually* configure two ports to join an EtherChannel:

Switch(config)# *interface range gi2/23 - 24*
Switch(config-if)# *channel-group 1 mode on*

The remote switch must also have the EtherChannel manually configured as ***on***. Remember that speed, duplex, VLAN, and STP configuration must be configured identically across all participating ports on *both* switches.

The *channel-group* number identifies the EtherChannel on the local switch. This number does not need to match on both switches, though for documentation purposes it *should*.

Adding switch ports to a channel-group creates a logical **port-channel interface**. This interface can be configured by referencing the *channel-group* number:

Switch(config)# *interface port-channel 1*
Switch(config-if)#

Changes made to the logical port-channel interface are applied to *all* physical switch ports in the channel-group:

Switch(config)# *interface port-channel 1*
Switch(config-if)# *switchport mode trunk*
Switch(config-if)# *switchport trunk allowed vlan 50-100*

To configure a port-channel as a Layer-3 interface:

Switch(config)# *interface port-channel 1*
Switch(config-if)# *no switchport*
Switch(config-if)# *ip address 192.168.10.1 255.255.255.0*

By default, a port-channel interface is *administratively shutdown*. To bring the port-channel online:

Switch(config)# *interface port-channel 1*
Switch(config-if)# *no shut*

Physical port properties, such as speed and duplex, must be configured on the physical interface, and not on the port-channel interface.

* * *

EtherChannel – Dynamic Configuration

Cisco switches support two dynamic aggregation protocols:

- **PAgP (Port Aggregation Protocol) –** Cisco proprietary aggregating protocol.
- **LACP (Link Aggregation Control Protocol)** – IEEE standardized aggregation protocol, originally defined in 802.3ad.

Both PAgP and LACP exchange **negotiation packets** to form the EtherChannel. When an EtherChannel is configured *manually*, no negotiation packets are exchanged.

Thus, an EtherChannel **will never** form if one switch manually configured the EtherChannel, and the other switch is using a dynamic aggregation protocol.

PAgP and LACP are **not compatible –** both sides of an EtherChannel must use the *same* aggregation protocol.

EthernChannel - PAgP

PAgP is a Cisco-proprietary aggregation protocol, and supports two modes:

- **Desirable** – actively attempts to form a channel
- **Auto** – waits for the remote switch to initiate the channel

A PAgP channel will form in the following configurations:

- desirable ←→ desirable
- desirable ←→ auto

A channel **will not form** if both sides are set to **auto.** Also, PAgP will not form a channel if the remote side is running LACP, or manually configured.

To create an EtherChannel using PAgP negotiation:

Switch(config)# *interface range gi2/23 – 24*
Switch(config-if)# *channel-protocol pagp*
Switch(config-if)# *channel-group 1 mode desirable*

Switch(config-if)# *channel-group 1 mode auto*

PAgP requires that speed, duplex, VLAN, and STP configuration be configured identically across all participating ports.

* * *

EtherChannel - LACP

LACP is an IEEE standard aggregation protocol, and supports two modes:

- **Active** – actively attempts to form a channel
- **Passive** – waits for the remote switch to initiate the channel

An LACP channel will form in the following configurations:

- active ←→ active
- active ←→ passive

A channel **will not form** if both sides are set to **passive.** Also, LACP will not form a channel if the remote side is running PAgP, or manually configured.

To create an EtherChannel using LACP negotiation:

Switch(config)# *interface range gi2/23 – 24*
Switch(config-if)# *channel-protocol lacp*
Switch(config-if)# *channel-group 1 mode active*

Switch(config-if)# *channel-group 1 mode passive*

LACP requires that speed, duplex, VLAN, and STP configuration be configured identically across all participating ports.

Recall that a maximum of **8 active ports** are supported in a single EtherChannel. LACP supports adding an *additional* 8 ports into the bundle in a **standby** state, to replace an active port if it goes down.

LACP assigns a numerical *port-priority* to each port, to determine which ports become active in the EtherChannel. By default, the priority is set to **32768**, and a **lower** priority is preferred. If there is a tie in *port-priority*, the lowest port number is preferred.

To change the LACP *port-priority* to something other than default:

Switch(config)# *interface range gi2/23 – 24*
Switch(config-if)# *lacp port-priority 100*

LACP also assigns a *system-priority* to each switch, dictated which switch becomes the decision-maker if there is a conflict about active ports. The default system-priority is **32768,** and a **lower** priority is again preferred. If there is a tie in *system-priority,* the lowest switch MAC address is preferred.

To globally change the *system-priority* on a switch:

Switch(config)# *lacp system-priority 500*

* * *

Troubleshooting EtherChannel

To view status information on all configured EtherChannels:

Switch# *show etherchannel summary*

```
Flags:      D - down        P - in port-channel
            I - stand-alone s - suspended
            R - Layer3      S - Layer2
            U - port-channel in use

Group       Port-channel      Ports
----------- --------------- ---------------
1           Po1(SU)           Gi2/23(P) Gi2/24(P)
```

Note that both ports have a status of *P*, which indicates that they are up and active in the EtherChannel.

On Cisco Nexus switches, the syntax for this command is slightly different:

NexusSwitch# *show port-channel summary*