CITES
Campus Information Technologies and Educational Services
University of Illinois at Urbana-Champaign



# ETHERNET AUTO-SENSING AND AUTO-NEGOTIATING

This page contains information about Ethernet auto-sensing and auto-negotiating for campus IT pros.

**Abstract**

This document, primarily intended for University of Illinois at Urbana-Champaign system administrators, explains the different ways that varying Ethernet standards communicate with each other and select a common speed and method of data exchange. The options discussed most thoroughly in this document are 10 megabits per second (Mbps), 100 Mbps, half-duplex, and full-duplex.

In addition, this document also discusses the configuration methods recommended for use on the Urbana-Champaign campus, and contacts for more information.

ON THIS PAGE



## INTRODUCTION

These days, twisted pair Ethernet comes in many different flavors. Not only are there different speeds available, there are also different duplex modes. Some Ethernet interfaces can run *full-duplex*, allowing information to be transmitted in both directions at the same time. This effectively doubles the available bandwidth on well-balanced links. In other cases, network topology or older interfaces may limit a network to running *half-duplex*, which allows a device to either transmit or receive, but not both at the same time.

If every Ethernet device could communicate using only one specific mode, it would be a networking nightmare. To ease this logistic problem, most newer interfaces that are capable of running the newer, faster, more robust configurations can also run the older, slower, more traditional ones. This allows a gradual upgrade of the network, one component at a time.

The problem remains that if you plug a device into a jack, the device needs to determine what mode it needs to run in to get the best performance. One way to do this would be to manually configure both sides of the connection to use a specific speed and duplex mode. The designers of the newer Ethernet specs knew this would not be a popular option, so they designed a system called *auto-negotiation* (sometimes refereed to by the misleading name *auto-sensing*), whereby an Ethernet interface can probe and determine the capabilities and characteristics of the other side of the new link. Once each side knows what the other side can do, they can pick the best communication method.

## AVAILABLE MODES

Before we get into how auto-negotiation works, we will look at the possible modes. Currently there are standards for three Ethernet speeds: 1000 megabits per second (also known as *Gigabit Ethernet*, or simply *GigE*), 100 megabits per second (also know as *Fast Ethernet*), and the original 10 megabits per second (sometimes refered to as *Legacy Ethernet*).

Gigabit Ethernet is still fairly expensive, and not all that commonly used on desktop machines. Fast Ethernet has become the de facto interface found on workstations. There are also millions of older hosts and networking gear and media that still run at the traditional 10 Mb.

With most newer computers having Fast Ethernet interfaces, that leaves two common choices in speed: 100 megabits per second, and 10 megabits per second. Add to that the ability to run half-duplex or full-duplex at each speed and we are left with four most common modes. Listed here from "best" to "worst," they are:

- 100Mb full-duplex
- 100Mb half-duplex
- 10Mb full-duplex
- 10Mb half-duplex

The auto-negotiation mechanism allows the two interfaces on a link to select the best common mode automatically, the moment a cable is plugged in. The problem is that it looks great on paper, but it doesn't always work as intended. Although the final Fast Ethernet standard did contain a section on auto-negotiating, that section was one of the last things put into the standard and many vendors had already implemented their own auto-sensing systems and deployed them before the standard was ratified. If this wasn't bad enough, there is no standard for detecting modes at 10Mb. As a result, there are a number of different ways various interfaces attempt to detect the capabilities of the link, and most of them are not compatible.

## HOW IT'S SUPPOSED TO WORK (AND WHY IT SOMETIMES DOESN'T)

First, let us get into how auto-negotiation is supposed to work before we get into why it often fails (or, more often, *appears* to fail). Let us assume we have two interfaces we want to connect together. Each interface has auto-negotiation enabled. When the interfaces first detect link with each other, each interface transmits a list of possible modes it can operate at. Once that is done, each interface knows both what it is capable of doing and what the other side is capable of doing. With this information, each side can independently pick the best common mode, and sets itself to use that mode.

That's how it is supposed to work, anyways. The most obvious problem is that the interfaces never actually communicate which mode they are setting themselves to. It is assumed that if both interfaces have the same lists, they will come to the same conclusion. That is usually true when both interfaces are using the same auto-negotiation system, but it isn't perfect. A bigger problem is when one of the interfaces doesn't follow the standard for auto-negotiation, and uses some other means to guess at the abilities of the other interface.

Auto-negotiation is an *active* method of determining link mode. Each interface is expected to transmit specific information in a specific format. If an interface that is expecting to use auto-negotiation does not receive this information from the other side, it assumes the other side cannot detect or change its mode. Many of the "pre-standard" detection means are *passive*. They looked at the characteristics of the system data they are sending and receiving to make a "best guess" at what the other side of the link is capable of doing. When using this passive auto-sensing mechanism, there is no communication between the interfaces regarding the mode at which they are to operate. Passive systems are not very reliable, which is why the Fast Ethernet standard uses auto-negotiating instead of auto-sensing. The problem is that when they mix, the passive system might guess correctly, but the active system will not receive the information it requires, and

falls back to modes that are usually less desirable.

In order to understand why this is happens, we must look at the problem these auto-negotiation and auto-sensing systems are trying to solve. The four modes come from two questions, each with two answers. The first question is "what speed?" with the choices of 10Mb or 100Mb. The second question is "what duplex?" with the choices of half or full.

Passively detecting speed is the easy one. Each Ethernet *frame* (or *packet*) starts out with a sequence of bits that alternate between 1 and 0 that looks like this: `1010101010101010...`. Each value (1 or 0) is represented by a specific state change, so when these bits are transmitted, the electrical signal on the Ethernet media transitions from "high" to "low" and back at the same speed the bits are being transmitted. To determine the speed, the interface needs to measure only the time between the transitions. If an interface is not capable of doing a higher speed, the bit pattern will look like signal noise, just like human speech played at ten times the normal speed sounds like noise. If each interface starts at its highest speed and works down, it can sync to the first speed it understands from the other side. This passive system allows the interfaces to determine a common speed very quickly with a great deal of reliability. It is also worth pointing out that the contents and format of the data that is sent is irrelevant, just the fact that the data is sent.

The task of passively determining duplex mode is almost impossible. There are ways to make a "good guess", but there is no way that you can be sure. To make things more complex, network topology can limit what duplex mode you can use. For example, if you have four machines that can each do full-duplex, but they are connected together using a *repeater*, they cannot operate in full-duplex mode. (Repeaters can also be called *hubs*, although not all hubs are strictly repeaters.) To make this task even more complex, the Ethernet link will still operate if the duplex modes are mismatched. If there is a speed mismatch, the link will not transfer data, but if there is a duplex mismatch, most of the data will still transfer correctly (especially on a low-usage machines or single-user machines), but one side of the link will have an abnormally high number of errors.

If there is a duplex mismatch, and one end of the link is in full-duplex mode when the other end is in half-duplex mode, network problems will occur. The half-duplex side of the link will not transmit data while it is still receiving data from the other side. The full-duplex side of the link has no such restriction, however, and will transmit the instant data is available to send-- it will not wait for the other side to stop transmitting before sending its own data. If the full-duplex side does this, the half-duplex side will start to receive data before it is finished with its own transmission. Seeing data on "both sides of the line," the half-duplex interface assumes a *collision* has taken place, discards the incoming packet, and counts it as an error.

This means the only way to detect, or attempt to guess, if the other side of a link can do full-duplex or not is to start transmitting something as soon as you start to receive a signal from the other end. The other side will start to receive your transmission before finishing up their own. If the other side is happy with this, it must be in full-duplex mode. If the other side thinks a collision has taken place, you know the other interface is in half-duplex mode.

**BUT**-- there is a catch. Almost all types of collisions are passive errors! If a collision happens, all the interfaces involved are expected to detect the error and react **independently**. That means that if a full-duplex interface is trying to probe an interface on the other side of a link by sending data "on top" of incoming data, even if the other interface is indeed half-duplex, senses a collision error and reacts, the original full-duplex interface never knows if the other interface considered the transmission an error or not! The end result is that there is a somewhat reliable (although not perfect) system of announcing the ability to do full duplex, but there is no way to probe the other side of the link.

Due to the problems with the older auto-sensing schemes (and the less than perfect ability of auto-negotiation to get things correct), many people have gotten in the habit of "forcing" an interface into a specific mode. In general, it is standard practice here at the University of Illinois U-C campus to hand configure all switch uplink interfaces and router interfaces to a specific mode of operation, and not rely on any of the auto-negotiating or auto-sensing systems. When a specific

mode is configured, it disables auto-negotiating and just blindly runs in that mode, no matter what the other side is doing.

### UNEXPECTED CONSEQUENCES (AND WHY THERE'S NO BETTER ALTERNATIVE)

This sometimes has unexpected results, however. Many people think that if they set a switch port (or interface) to a specific mode (say 100Mb/full-duplex), and then plug in a workstation, it should automatically sync up with the switch port and configure itself to 100Mb/full-duplex. They are then surprised when the workstation settles into 100Mb/**half**-duplex, and their network starts experiencing a very high number of errors. The problem is that most new interfaces shipping today (including most switch interfaces) use the more robust auto-negotiation standard. As stated before, auto-negotiating is an active system, requiring data be sent by both sides. **If auto-negotiation is turned off at the switch, any host plugged into it will send its auto-negotiation information and wait for the switch to send its own auto-negotiation information back. When the switch fails to do so (because auto-negotiation is disabled), the host assumes the interface is not capable of full-duplex. It falls back on the passive means used to detect speed, and sets itself up at the correct speed, but in half-duplex mode.**

Although this may look dumb at first, it really is the Right Thing To Do. When the host failed to get any information from the switch interface, it has no idea what it could or could not do. Since there is no good way to detect duplex mode, it assumed the lowest common denominator (half-duplex). More importantly, it had no idea what the network topology between it and the switch looks like. If the switch port was connected to a hub that had many workstations plugged into it, the host would cause serious errors for the whole network segment if it put itself in full-duplex mode.

### THE UPSHOT: EITHER TRUST AUTO-NEGOTIATION OR MANUALLY CONFIGURE EVERYTHING

What this all boils down to is this: you have to make a choice of either trusting auto-negotiation to get it right, or you have to hand configure ***both*** sides of the Ethernet link. This is especially true if you intend to use full-duplex mode. If either side of the link has auto-negotiation turned off, the other side of the link will find the right speed, but will always default to half-duplex. If you want full-duplex, you must leave auto-negotiation enabled on both sides of the link, or you must hand configure both sides for full-duplex.

This can be a pain on some systems, such as laptops, that are moved around from port to port. To make the situation even worse, many vendors and OS makers don't provide a means to force the interface to one mode or another-- you have to trust auto-negotiation because it is the only game in town. In most cases, this is not a problem. Auto-negotiation is considerably better than auto-sensing at getting everything correct, and even if there is a duplex mismatch, the average workstation will not generate enough errors to cause concern. If worse comes to worse, you can just unplug the cable for a few seconds, and then plug it back in. The interfaces will reinitialize themselves, and go through the auto-negotiation sequence again.

Where many problems arise is with servers or multi-user systems that have a large amount of network I/O. Luckily, most of these systems tend to be rather static. They tend not to move around a lot, and they tend to have many manual configurations and tweaks to get the best performance. One more network configuration (if it is available) isn't a problem.

It should also be noted that 100 Mb interfaces that were manufactured when the 100 Mb standard was first ratified (or before) often have serious problems interoperating with systems that follow the full auto-negotiation standard. There have been cases of systems from this era refusing to link at all until the switch interface it was connected to had a specific duplex and speed mode configured. If you have one of these interfaces manufactured in the mid 90s, extra care may be required to configure proper speed and duplex modes to get the system to work at all with the newer standardized hardware.

I hope this has helped you understand more about auto-negotiation. If you are a member of the

UIUC campus community and have a question about your own network configuration, please see your department's Network Administrator. If you **are** the Network Administrator, and still have questions, please contact the Network Administrator Support group (NAS) at CITES. They should be able to answer general questions, and help understand any auto-negotiation issues you might have. NAS is also a good source of information for when it might (or might not) be a good idea to hand-configure switch port speeds and duplex settings. If NAS is unable to help, you can also contact the Network Design Office (NDO) at CITES. They should be able to answer questions about specific networking hardware, or route any additional questions.

## QUICK SUMMARY

- Auto-negotiation is not 100% reliable, but it does generally work.
- Links with a duplex mismatch will operate, but will generate large numbers of errors, and can slow down busy networks.
- For most interfaces, both speed and duplex need to be set to auto for full auto-negotiation to work.
- Forcing a Catalyst switch port to a specific speed disables auto-negotiation for the duplex setting.
- Full-duplex mode can be achieved only if **both** sides of the link are **either** set to auto-negotiate **or** manually configured to use full-duplex.
- Full-duplex will work only if a host is connected directly to a switch or other device, with no repeaters or hubs in-between.
- If auto-negotiation is enabled on only one side of the link, it will **always** default to **half**-duplex, regardless of what the other side of the link is forced to.
- If one side of a link is forced to full duplex and the other is set to auto-negotiation, a duplex mismatch will occur.
- You can force a new auto-negotiation by simply unplugging a host cable for 10 seconds.
- Most 10 Mb interfaces can run only in 10Mb half-duplex mode.
- Most 10/100 Mb interfaces can do auto-negotiation. Most 10/100 Mb interfaces with RJ-45 twisted pair jacks can run in full-duplex mode.
- Any network connected via an AUI port (with an external transceiver, for example) can run only in 10Mb half-duplex mode.

## MORE INFORMATION

If you have a machine with a 10/100 or 10/100/1000 Mb interface and need to force a specific operating mode, have a look at these resources. If your machine type is not shown, try checking with the manufacturer for more information. This page will be updated with any new information as it becomes available. Please feel free to email us with any additional information you might have.

### Sun SPARC systems

The easiest way to configure a Sun Microsystems computer with a "hme" Fast Ethernet controller is with hmeconfig. This includes the on-board Ethernet controller for all "Ultra 2" systems and newer, along with SBus and PCI Fast Ethernet NICs. That does not, however, include Sun's "Quad Fast Ethernet" NIC. To find out if your workstation has an hme controller, issue the command `netstat -i`, and look for something like `hme0` under the `name` column.

### IBM AIX systems

SMIT happens. Run `smit`, and go through the following menus: "Devices" to "Communications" to "Ethernet" to "Adapter" to "Change / Show Characteristics of an Ethernet Adapter." You should be presented with options to change the speed and duplex, if your host supports these options. I've also been told some RS/6000 system have PROM options to change the Ethernet mode.

**Apple Macintosh Pre-OS X:**
**Different pre-OS X Macintoshes need different versions of duplex tools. See http://www.appletechs.com/archives/00000044.html for a discussion of which version of the tools are needed for which Macs, and for download locations. -->**

**OS X 10.2 or later:**

Versions of OS X between 10.0 and 10.1.x did not completely support all the duplexing options available. However, from 10.2 forward, you can use the `ifconfig` command to set both the duplexing setting and the speed.

To display the current Ethernet settings, use the Terminal application to run the `ifconfig` command with no arguments.

To change to full duplex or half duplex, use:

```
sudo ifconfig en0 mediaopt full-duplex
```

or

```
sudo ifconfig en0 mediaopt half-duplex
```

To change speed, first make sure which speed your network connection supports. Then check the output of ifconfig to see what Ethernet options are supported by your network card. Then use a command like:

```
sudo ifconfig en0 media 10baseT/UTP mediaopt full-duplex
```

(or half-duplex)

or

```
sudo ifconfig en0 media 100baseTX mediaopt full-duplex
```

(or half-duplex, etc.)

See the `ifconfig` man page for more information.

**Microsoft Windows 9*/ME/NT/2000**

Windows systems can configure their Ethernet controllers from the Network Control Panel. You can find the Network Control Panel in the Control Panel folder under "Settings" in the Start Menu. The "Ethernet Adapter Properties" panel can be found by clicking on the adapter under the "Configuration" tab in Windows 9x, or under the "Adapters" tab in NT 4. Options to set speed and duplex are usually found under an "Advance" tab.

You can also get there by right-clicking on the Network Neighborhood or My Network Places and selecting Properties. (In Windows 2000 you would then right-click on Local Area Connection and select Properties again.)

Under Windows 2000, and depending on the driver in other flavors of Windows as well, you can also modify these settings from the Adapter Properties in the Device Manager. The Device Manager can be found in the System Control Panel or System Properties, or by right-clicking on "My Computer" and selecing Properties. In Windows 9x the Device Manager is a tab. In Windows 2000 it is under the Hardware tab.

**Cisco Catalyst Switches**

## Departmental Switches

On most multi-module Cisco switches, you can display the current port status with the command: `show port` *`mod/port(s)`* The *Duplex* and *Speed* columns of the table will tell you what the current

port mode is set to. If the port has been forced to a specific mode, that mode will simply be listed. If the port is set to auto-negotiation, but nothing is plugged into it, the speed and duplex will be reported as `auto`. If the current mode was the result of a successful auto-negotiation, the labels will be prefixed with "`a-`" (e.g. `a-100` or `a-full`). To change the mode of a port, you can use the following commands: `set port speed` *`mod/port(s) mode`*, with *mode* being either `10`, `100`, or `auto`. `set port duplex` *`mod/port(s) mode`*, with *mode* being either `full` or `half`. If a port has its speed mode set to `auto`, the duplex mode will also be set to `auto` and you will not be able to change it. In order to change the duplex mode, you will need to set the speed mode to `10` or `100`. For more information, see the Cisco Catalyst 5000 series Ethernet/Fast Ethernet Module manual.

If you are depending on auto-negotiation, please be sure that `portfast` is configured `on`, and `etherchannel` should be set to `off`.

## Multi-VLAN Shared Switches

Department Net Admins do not have command line access to multi-VLAN switches. Net Admins can view configurations for switch ports on your VLAN with the IRIS web tool. IRIS also allows Net Admins to change port modes on most commonly used switches.

### CONTACTS (FOR UIUC AFFILIATES)

For assistance or questions with IRIS, contact the Network Administrator Support group (NAS) or admin-help@uiuc.edu.

If your multi-VLAN switch is not yet supported by IRIS, your network administrator will need to contact the Operations Center at net-trouble@uiuc.edu. They will be happy to set a port to a specific mode. Please have the port number(s) you wish to change and the mode you wish them set to before you contacting your designer.

---

Written by Jay Kreibich on Mon. Oct 16, 2000.

Ported to the CITES website by CITES Documentation on Oct. 29, 2002.

Last modified September 02 2008