

# **ETHERNET/IP ADDRESSING ISSUES AND RECOMMENDATIONS**

Jamin D. Wendorf  
Senior Design Engineer  
Real Time Automation  
2825 N. Mayfair Rd. Suite 11  
Wauwatosa, WI 53222  
[jwendorf@rtaautomation.com](mailto:jwendorf@rtaautomation.com)

## **KEYWORDS**

DHCP, BOOTP, ARP, Gratuitous ARP, “Fast DHCP”, “Fast BOOTP”

## **ABSTRACT**

Some of the obstacles to widespread acceptance of Ethernet on the factory floor are device replacement and IP address assignment. In the office environment replacing a printer may mean that some PCs are unable to access the printer until a server is rebooted or some other manual action is taken. Manual intervention of this sort is highly undesirable on the factory floor. In a factory environment it is very desirable for the IP Address of a new device to have the same IP Address as the device it replaced. In the ideal case the IP Address of the new device would be set automatically without the intervention of an operator.

The paper discusses device addressing as described by the Ethernet/IP specification, the optional methods outlined in the specification and the new strategies under consideration Ethernet/IP Workshop/System JSIG. The paper also touches on other issues surrounding this topic including Duplicate IP Address Detection and Defense.

This paper presents issues and strategies under active discussion by the EtherNet/IP Implementer Workshop and the EtherNet/IP System JSIG. Permission to present this information was granted by the chair.

## INTRODUCTION

It's 2:00am on the weekend. Bob, your service technician, makes his normal rounds and finds that one of your EtherNet/IP™ devices has failed. Bob knows the device needs to be replaced in a hurry to prevent your company from losing thousands of dollars. Armed with a screwdriver and the replacement device, Bob heads to the failed device to replace it. Only there is a problem. Bob needs to make sure the new device can talk on the existing network. With DeviceNet™ Bob only needed to set an 8-bit address with few switches and everything worked. But EtherNet/IP uses a 32-bit address. What is Bob to do?

This is just one of the issues surrounding IP addresses that EtherNet/IP hasn't standardized. There are many different approaches a developer can take when replacing a device to ensure the new device looks identical to the old device. Below are a few configuration methods a developer can include in their device to configure the new device:

- Serial configuration
- Removable EPROM, Flash Stick or Memory Button
- Rotary or dip switches
- LCD and Keypad
- "Smart Switch" that remembers the IP Address of the device plugged into particular physical port
- BOOTP/DHCP

As you can see, the problem isn't how to configure your device. The real problem is how to configure your device in a standard way. The EtherNet/IP Developer's Workshop has been working on this issue for the better part of a year. This document covers the problems and recommendations surrounding IP Addressing including initial configuration, device replacement, duplicate IP Address detection and IP Address defense. The recommendations are just that at this point. If the recommendations are accepted by the workshop, the next step is to draft an EtherNet/IP Specification Enhancement (ESE) for approval by the Technical Review Board (TRB) and the EtherNet/IP System JSIG.

### **General Addressing Methods**

Bootstrap Protocol (BOOTP) is a protocol that lets a network user be automatically configured (receive an IP Address) and have an operating system booted (initiated) without user involvement. The BOOTP server, managed by a network administrator, automatically assigns the IP address from a pool of addresses for certain duration of time. BOOTP is the basis for a more advanced network manager protocol, the Dynamic Host Configuration Protocol (DHCP).

DHCP is a communications protocol that allows network administrators to manage and automate the issuing of IP Addresses within a Local Area Network (LAN). Without

DHCP, all IP addresses must be entered manually at each device. If the device moves to another subnet, a new IP address must again be manually entered. DHCP uses the concept of a lease to prevent the loss of an IP address to a device which was removed.

DHCP is based on BOOTP and maintains some backward compatibility. The main difference between DHCP and BOOTP is the amount of user involvement. BOOTP was designed for manual pre-configuration of the host information in a server database. DHCP allows for dynamic allocation of network addresses and configurations to newly attached hosts. In addition, the use of the lease allows DHCP to reallocate and recover when the network has problems.

Devices may also be configured manually. This is convenient for devices that need static addresses that don't want to support BOOTP. Devices that don't support BOOTP or DHCP must provide a way to configure the IP address. This is commonly done using a proprietary serial communication protocol. This is effective for static IP address configuration, but makes replacing a device very complicated.

### **Initial Configuration**

Most, if not all, EtherNet/IP clients require a static IP Address for communications. That being said, the IP Address still needs to be configured initially. It is recommended by the EtherNet/IP Developer's Workshop that all devices support DHCP and/or BOOTP (described later) for IP address configuration. When a device has no IP Address, the only way to talk to the unit over Ethernet is to communicate through low-level protocols using the device's MAC address (6 byte worldwide unique number). When a device has no IP Address, the *Configuration Control* attribute (Class 0xF5 TCP/IP Interface Object, Instance 1, Attribute 3) should default to DHCP or BOOTP. On power up, the device should issue a DHCP or BOOTP request. A simple User Interactivity DHCP Server Utility captures the request and prompts the user to enter an IP address. After the IP address is set, save the address to non-volatile memory and change the *Configuration Control* attribute to use the values stored in non-volatile memory. See Table 1.

<b>Bit(s):</b>	<b>Called:</b>	<b>Definition</b>	
0-3	Startup Configuration	Determines how the device shall obtain its initial configuration at start up.	<p>0 = The device shall use the interface configuration values previously stored (for example, in non-volatile memory or via hardware switches, etc).</p> <p>1 = The device shall obtain its interface configuration values via BOOTP.</p> <p>2 = The device shall obtain its interface configuration values via DHCP upon start-up.</p>

Bit(s):	Called:	Definition
		3-15 = Reserved for future use.
4	DNS Enable	If 1 (TRUE), the device shall resolve host names by querying a DNS server.
5-31	Reserved	Reserved for future use and shall be set to zero.

**Table 1 - Configuration Control Attribute**

### **Normal Power Up**

During a normal power up, a device should read its address out of non-volatile memory then send a Gratuitous ARP message to see if any other device is using its IP Address (See Duplicate IP Detection and IP Address Defense below). Upon successfully validating no other device is using the IP Address, the device goes online. This option works, but doesn't allow the IP Address to be changed without modifying the *Configuration Control* attribute.

The EtherNet/IP Developer's Workshop is suggesting a "Fast DHCP" and "Fast BOOTP" option of the *Configuration Control* attribute. "Fast DHCP" and "Fast BOOTP" would send a DHCP/BOOTP request on every power up to allow the IP Address to be modified, but only wait a couple of seconds (standard DHCP/BOOTP suggests waiting for up to 60 seconds if no server is available) for a DHCP/BOOTP server before giving up and going online with a saved IP Address. This allows a device to use the same IP address on every power up, even if no DHCP/BOOTP server is available, while still allowing the address to be modified if necessary.

### **Duplicate IP Address Detection**

All EtherNet/IP, and more generically Ethernet, devices must be assigned a unique address within their subnet. Ethernet doesn't provide a common standard for Duplicate IP address detection or defense, but there is a need, especially in the factory. Stuart Cheshire from Apple Computers has suggested the following method which suggests using DHCP with Address Resolution Protocol (ARP).

The DHCP specification (RFC 2131) briefly discusses the role of ARP in validating configuration as follows. Upon receiving an IP address, the DHCP client should probe the newly received address by sending an ARP. If no other devices are using the new address, the DHCP client can use it. If another device is using the new address, the DHCP client must send a *DHCPDECLINE* message to the DHCP server and try a new address.

The main problem with this method is the vagueness of the behavior. The DHCP RFC didn't specify the number of ARP packets to be sent, the interval between packets, or the time frame to wait before assuming the address isn't in use. Stuart suggests each DHCP client waits a random time interval from 0-200 milliseconds with four probe packets,

waiting 200 milliseconds after each probe for a maximum of 800–1000 milliseconds before a chosen IP address may be used. Duplicate IP Address Detection shouldn't be limited to power up though. Stuart suggests all devices continually look for ARP requests and responses where the sender's IP Address matches their IP Address.

### **IP Address Defense**

Upon receiving an ARP packet to its IP address, a device must know when to back off and request a new IP Address and when to defend its IP Address in an attempt to have the other device back off. The main problem with IP Address Defense is determining priority. It is suggested the existing device should back off for the new device when their IP Addresses match. Most people would agree a device with active TCP/IP connections shouldn't stop communicating if a new device wants to use its IP Address.

If you don't want to back off, you may try to defend your address by sending a broadcast ARP with your configuration. This tells the other device you are keeping the IP address. The other device must then back off or send you a message stating his intention to keep using the IP address. As you see, one device must stop trying to use the IP address, or this defense scheme never ends.

As of right now, this is an issue that hasn't been resolved. No matter what priority and protocol is involved, there is always the case of two devices that are configured with the same behavior. The only solution at this time is to use some form of TCP/IP status reporting (telnet dump, SNMP, web server) to alert the user of the problem and let them decide which device is to use the IP address.

The biggest problem with Duplicate IP Address and IP Address Defense is the lack of protection for the Device that not only has the IP Address, but is also communicating. Right now a malicious device can stop an active device from communicating by send an ARP request to its IP address. This is still a big issue with the EtherNet/IP Implementer's Workshop.

### **Automatic Device Recovery (ADR)**

Automatic Device Recovery is a serious issue in the factory. When a device fails, no one wants to wait minutes much less hours for the line to recover. Many approaches have been discussed surrounding this issue. The key point is to develop a method for replacing a device that doesn't involve your IT department. Ideally the approach would be done without the need for a PC at all.

Both DHCP and BOOTP require user intervention to work. DHCP typically matches the device MAC address to a predefined IP Address or issues an IP Address from a reserved range. BOOTP typically requires immediate user input to assign an IP address to the device. Both methods can be configured to assign the same address to the same MAC address on every request. This is a nice feature during normal conditions. The problem occurs when a device fails. In order to replace the device the DHCP/BOOTP server

needs to remove the old MAC address from its lookup table and replace it with the new MAC address, as of right now a manual task.

A second approach was discussed that involves a removable memory device (flash stick, EPROM, etc...) that contains the configuration of the device including the IP Address. This would solve a lot of problems. Now when a device fails, the user can remove the memory stick, swap the device and replace the memory stick. This gets rid of one problem. We now need to inform the DHCP/BOOTP server that the old MAC address is no longer in use. If we had a smart DHCP/BOOTP server, we could look at the request, see it is for a different devices MAC address and ARP the old MAC address to validate it is no longer on the network. This approach has cost as a downfall, since the memory sticks consume board real estate and extra parts.

The last approach would only work is a single device fails. A smart DHCP/BOOTP server can recognize the loss of a device and the addition of the new device, assign a temporary IP to validate *Identity Object* attributes to prove the new device is the same type as the old device, then assign the real IP address and set the IP address to static. This allows the technician replacing the device to simply remove the old device and add the new device. Again, this would only for a single failure of a particular device (If two photo eyes from the same manufacturer fail, how would a smart DHCP/BOOTP distinguish between the two?)

## CONCLUSION

While many standard methods are currently in place to address the issue of IP Address configuration, there still isn't a standard way to assign an IP Address over EtherNet/IP. The recommendations in general are as follows:

- Support “Fast DHCP” or “Fast BOOTP” for static address use with the option to change the address by the DHCP/BOOTP server.
- Support Gratuitous ARP for Duplicate IP Address Detection.
- Fight for your IP Address if you are already communicating.
- Back off if your try to use an IP Address already in use by another device.

If all devices support the above recommendations, EtherNet/IP moves even closer to standardized means of configuration and address troubleshooting.