

# **IPv4 Address Conflict Detection for EtherNet/IP Devices**

**Version: 1.5**

**6-June-2006**

## Contents

1	Introduction.....	4
1.1	Scope of Document.....	4
1.2	Problem Description .....	4
1.3	Overview of the Solution.....	5
2	Cheshire IPv4 ACD Mechanism [informational] .....	6
2.1	Summary .....	6
2.2	Initial Address Probing .....	6
2.3	Address Announcement.....	6
2.4	Ongoing Conflict Detection and Defense .....	6
2.5	IPv4 ACD Timing Constants.....	7
2.6	IPv4 ACD Status and Issues .....	7
3	IPv4 ACD Adaptation for EtherNet/IP Devices .....	8
3.1	General.....	8
3.2	Timing Constants .....	8
3.3	Initial Address Probing .....	8
3.4	Address Announcement.....	9
3.5	Ongoing Conflict Detection and Defense .....	9
3.6	Additional Requirements .....	10
3.7	EtherNet/IP Fault Actions.....	10
3.8	Summary of IPv4 ACD Behavior for EtherNet/IP Devices .....	10
4	Implementation Considerations [Informational].....	13
4.1	Summary .....	13
4.2	Implementation Requirements .....	13
4.3	Implementation Example .....	13
4.4	Example Pseudo-random Delay Algorithm .....	13
4.5	Issues With DHCP Clients.....	14
4.6	Future Direction .....	14
5	Idiosyncrasies Observed with IPv4 ACD Behavior [Informational].....	15
5.1	Summary .....	15
5.2	Infinite DHCP Client Loop.....	15
5.3	Switches Can Drop ARP probes .....	15
6	Test Cases [Informational].....	16
6.1	Summary .....	16
6.2	Tests with Other IPv4 ACD Device.....	16
6.3	Tests with Devices Not Supporting IPv4 ACD .....	18
7	References.....	19
	Appendix A – IPv4 Address Conflict Detection Internet Draft.....	20

<b>Version:</b>	<b>Date:</b>	<b>Description:</b>
1.0	27-July-04	First release.
1.1	26-Aug-04	Edits to test procedure; not officially released.
1.2	3-Jan-06	Minor edits; not officially released.
1.3	9-May-06	Updates for readability, and to make consistent with IAONA version. Significant changes from v1.0: <ul style="list-style-type: none"><li>• Inclusion of latest [Cheshire] Draft as appendix</li><li>• Addition of state diagram</li><li>• Clarification that ongoing periodic probes are recommended but not required</li><li>• Clarification of test cases</li></ul>
1.4	12-May-06	Clarified conflict behavior post-probing (with respect to DHCP behavior).
1.5	6-Jun-06	Modified use of “must”, “should”, etc. as requested by Safety; miscellaneous formatting.

## 1 Introduction

### 1.1 Scope of Document

This document specifies a common mechanism that EtherNet/IP devices can use to detect and act upon IPv4 address conflicts. The address conflict detection (ACD) mechanism is based upon, and conforms to, the Internet Draft authored by Stuart Cheshire of Apple (see [Cheshire05] in References).

In addition to specifying the use of the Cheshire IPv4 ACD mechanism, this document specifies additional requirements for EtherNet/IP devices with respect to the IPv4 ACD mechanism. Note that at this time the ACD mechanism is optional for EtherNet/IP devices.

An EtherNet/IP device implementing the ACD mechanism shall:

1. Implement the ACD mechanism specified in [Cheshire05].
2. Implement the additional requirements for EtherNet/IP devices (Section 3).

In addition to the ACD mechanism, this document provides informational material and test cases to assist developers in implementing the mechanism.

### 1.2 Problem Description

Unpredictable and undesirable behavior results when multiple devices attempt to use the same IP address. An example address conflict scenario might be shown below:

- Device A starts up and uses IP address 10.88.80.10
- Other devices establish communications with Device A
- Device B has incorrectly been configured to also use 10.88.80.10
- Device B starts up and sends an ARP announcement
- Existing communications to Device A are disrupted and reestablished with Device B (though this is not necessarily always the case)
- New communications initiated by other devices could go to Device A or Device B (or may be established and disrupted again)

In such situations, it can be difficult to determine that there even is an address conflict situation, and where the offending device is located.

To further complicate the problem, there is no universally implemented mechanism for detecting IP address conflicts. A Windows PC typically displays a pop-up message if it receives a response to ARP announcement messages sent at startup. Some embedded devices follow the Windows behavior, but in general there is no common behavior for embedded devices.

It is desirable to have a common IP address conflict detection mechanism for EtherNet/IP devices. A common mechanism enables consistent behavior and helps the user in diagnosing address conflict conditions.

### **1.3 Overview of the Solution**

The ACD mechanism for EtherNet/IP devices is based on the Internet Draft for IPv4 Address Conflict Detection (referred to as IPv4 ACD) developed by Stuart Cheshire of Apple. The most recent version of the IPv4 ACD draft is included as Appendix A in this document.

The IPv4 ACD mechanism involves the following aspects:

1. Initial probing: before using an IP address the device issues ARP Probes to see whether the address is in use.
2. Address announcement: after the initial probing the device issues ARP announcements.
3. Ongoing conflict detection and defense: the device performs ongoing address conflict detection and potentially defends its address.

In addition to specifying the use of the Cheshire IPv4 ACD mechanism, this document specifies additional requirements for EtherNet/IP devices with respect to the IPv4 ACD mechanism. Additional aspects include:

- a. Timing of ARP probes
- b. Address defense behavior
- c. Device fault behavior

## **2 Cheshire IPv4 ACD Mechanism [informational]**

### **2.1 Summary**

This section summarizes the essential aspects of the Cheshire IPv4 ACD mechanism as specified in the Internet Draft. For a full description of the mechanism, refer to the IPv4 ACD Internet Draft included in Appendix A.

### **2.2 Initial Address Probing**

Before using an IP address a host must determine if the address is already in use. Initial address probing must be done regardless of whether the address was obtained via BOOTP, DHCP, or via other means (e.g., manual configuration).

In order to determine whether an address is already in use, the host waits for an initial random delay, then issues ARP probes (an ARP request with the target IP address in question and sender IP address of all 0).

The use of ARP probes for the initial conflict detection is beneficial since hosts will not update their ARP caches upon receipt of an ARP probe. As a result, if a host is already using an IP address and a second host starts up and probes for the address, the communications to the original host will not be disrupted.

During initial ARP probing, if any ARP message is received with the Sender IP Address matching the address being tested, then a conflict has been detected. In addition, if an ARP probe is received with the Target IP Address matching the address being tested (and the Sender Hardware Address does not match any of the host's interfaces) then a conflict is detected.

### **2.3 Address Announcement**

After the initial address probing has been completed, and no conflict has been detected, the host must issue ARP announcements (ARP request with the target IP address and sender IP address set to the address in question) to announce its use of the address.

### **2.4 Ongoing Conflict Detection and Defense**

After initial address probing and announcement, hosts must perform ongoing conflict detection and defense. A conflict is detected if the host receives an ARP message (request or reply) with a Sender IP Address that matches the host's IP address. Note: ARP messages that contain the host's own hardware address in the Source Hardware Address field are NOT treated as conflicts.

When a conflict is detected after the initial probing, a host has 3 options:

- a. Cease using the address.
- b. Attempt to defend the address by issuing an ARP announcement, provided a conflict was not detected within the last DEFEND\_INTERVAL seconds (in order to prevent a persistent conflict/defend condition with another host that does not implement the ACD mechanism).
- c. Defend the address indefinitely if configured to do so. This might be the course of action taken by routers or other servers with fixed, permanent IP addresses. As

above, there is an additional requirement that a host defending an address not send defensive ARP announcements more frequently than every DEFEND\_INTERVAL seconds.

## 2.5 IPv4 ACD Timing Constants

The IPv4 ACD Draft specifies a number of timing-related constants. The constants are shown below, with their nominal values:

PROBE_WAIT	1 second	(initial random delay)
PROBE_NUM	3	(number of probe packets)
PROBE_MIN	1 second	(minimum delay until repeated probe)
PROBE_MAX	2 seconds	(maximum delay until repeated probe)
ANNOUNCE_WAIT	2 seconds	(delay before announcing)
ANNOUNCE_NUM	2	(number of announcement packets)
ANNOUNCE_INTERVAL	2 seconds	(time between announcement packets)
MAX_CONFLICTS	10	(max conflicts before rate limiting)
RATE_LIMIT_INTERVAL	60 seconds	(delay between successive attempts)
DEFEND_INTERVAL	10 seconds	(minimum interval between defensive ARPs)

Note: Since some of these values are not optimal for industrial networking devices, alternate values are specified in Section 3.

## 2.6 IPv4 ACD Status and Issues

Previous correspondence with the Draft's author indicated that the IPv4 ACD Internet Draft had been accepted by the IETF Zeroconf Working Group, and was waiting further disposition by the Internet Area Coordinator.

While it seems likely that the IPv4 ACD Draft will become an RFC, this is not guaranteed. EtherNet/IP devices can still implement the mechanism, per the Internet Draft (included in Appendix A of this document) and the adaptation requirements for EtherNet/IP devices. Indeed, the IPv4 ACD mechanism has been present for several years in Apple-developed operating systems.

Note: the ACD mechanism as specified here recommends that periodic ARP probes be sent as part of ongoing detection. [Cheshire05] states that "hosts Must Not perform this check periodically as a matter of course" (section 2.1). It has been demonstrated that the periodic probes are necessary to detect conflicts in some situation: for example, when a simple target is not connected to the larger network when powered up, or when switches drop initial ARP probes due to initial forwarding delay. It is hoped that the conflict with [Cheshire05] is resolved in a newer version of the document.

### 3 IPv4 ACD Adaptation for EtherNet/IP Devices

#### 3.1 General

In order to make the IPv4 ACD mechanism feasible for EtherNet/IP devices, and to achieve consistent behavior, it is necessary to specify a number of particular requirements with respect to [Cheshire05].

Unless otherwise specified, EtherNet/IP devices implementing this mechanism shall follow [Cheshire05]. In addition, devices shall follow the specific EtherNet/IP adaptations stated in the following sections.

#### 3.2 Timing Constants

[Cheshire05] specifies a number of timing-related constants, some of which are not optimal for EtherNet/IP applications. In particular, the start up delays cause by the ARP probe intervals would be unacceptable in the industrial setting.

The following list shows the alternate values that shall be used (constants not listed are unchanged from [Cheshire05]):

PROBE_WAIT	200 ms	(initial random delay)
PROBE_NUM	4	(number of probe packets)
PROBE_MIN	200 ms	(minimum delay until repeated probe)
PROBE_MAX	200 ms	(maximum delay until repeated probe)
ANNOUNCE_WAIT	200 ms	(delay before announcing)

In addition to the above constants, devices shall use the following constants when sending ARP probes as part of ongoing conflict detection:

ONGOING_PROBE_MIN	90 seconds	(minimum interval for ongoing probes)
ONGOING_PROBE_MAX	150 seconds	(maximum interval for ongoing probes)

#### 3.3 Initial Address Probing

For the initial address probing, EtherNet/IP devices shall do the following:

- Perform initial address probing before using an IP address – at power up and any other time the IP address is configured or changed. The IP address shall be probed regardless of whether the address was obtained via BOOTP, DHCP, or static configuration (see also section **Error! Reference source not found.**, Issues with DHCP Clients).
- Do not start the initial address probing until Ethernet Link Integrity has been detected.
- Devices are recommended to randomize the initial delay before probing in order to avoid a situation where many like devices power up and send ARP probes simultaneously. There are a number of ways to generate a pseudo-random number in the range of 0-PROBE\_WAIT (200 ms as specified above). A simple method is to perform arithmetic on the Ethernet address to produce an initial delay value (e.g., perform “modulo” arithmetic on the Ethernet address and use

the remainder as the initial delay in ms). Refer to Section **Error! Reference source not found.** for an example.

- If a conflict is detected during the initial probing, the device shall do the following:
  - The device shall not use the IP address.
  - Take the EtherNet/IP LED and device status actions as specified in Section 3.7.
  - If the address was obtained via DHCP, the device shall take one of two actions:
    1. Issue a DHCPDECLINE, remain in the address conflict state and do not restart the DHCP client state machine until the device is reset. This is the recommended action, since it addresses the situation where a misconfigured server continues to give out the same (fixed) address.
    2. Issue a DHCPDECLINE, then restart the DHCP client state machine to the Init state (issue DHCPDISCOVER). The device shall limit the rate at which new addresses are requested and probed, per [Cheshire05] (i.e., after MAX\_CONFLICT conflicts the device shall probe for new addresses no more than once per minute).
  - If the address was obtained via BOOTP, do NOT reissue BOOTP requests since the BOOTP server will simply reissue the same IP address.
  - If the device has other communications ports over which it can be configured, it is strongly recommended that the device allow the user to set a new, non-conflicting IP address over the other communication port(s).

### **3.4 Address Announcement**

There are no additional requirements beyond [Cheshire05].

### **3.5 Ongoing Conflict Detection and Defense**

After an IP address has successfully been probed and put into use, a device shall perform ongoing conflict detection and defense according to [Cheshire05], with the following specific requirements:

- If a conflict is detected, it is strongly recommended that the device attempt to defend its address per Option (b) in [Cheshire05]. However there are some circumstances where a device may elect not to defend its address. For example, if a mobile device detects a conflict, it is reasonable to assume the mobile device is the problem. In this case it is better for the mobile device to cease using the address.
- If a conflict persists (as defined by [Cheshire05]) the device shall follow the behavior as when a conflict is detected during initial probing (device state, LEDs, DHCP behavior, etc.), with one exception: if the address was obtained via DHCP, the DHCP client shall issue a DHCPRELEASE.

- It is recommended that devices issue periodic ARP probes within the range of ONGOING\_PROBE\_MIN and ONGOING\_PROBE\_MAX (“randomized” using Ethernet MAC address). The periodic ARP probes allow the module to detect conflicts with devices that may not have been connected to the network at the time of initial probing, or when switches have dropped the initial ARP probes due to initial forwarding delay.

### 3.6 Additional Requirements

- Devices shall respond to ARP probes. Note that some older TCP/IP implementations ignored ARP probes.
- When a device detects that Ethernet link integrity has been lost and then regained (e.g., the network cable has been removed and replaced), the device shall re-start the initial conflict detection behavior.

### 3.7 EtherNet/IP Fault Actions

When an address conflict is detected, during the initial probing or ongoing detection phases, EtherNet/IP devices shall take the following fault actions:

- Set the device state to Recoverable Fault (Module Status LED flashing red).
- Set the Network Status LED to solid red.

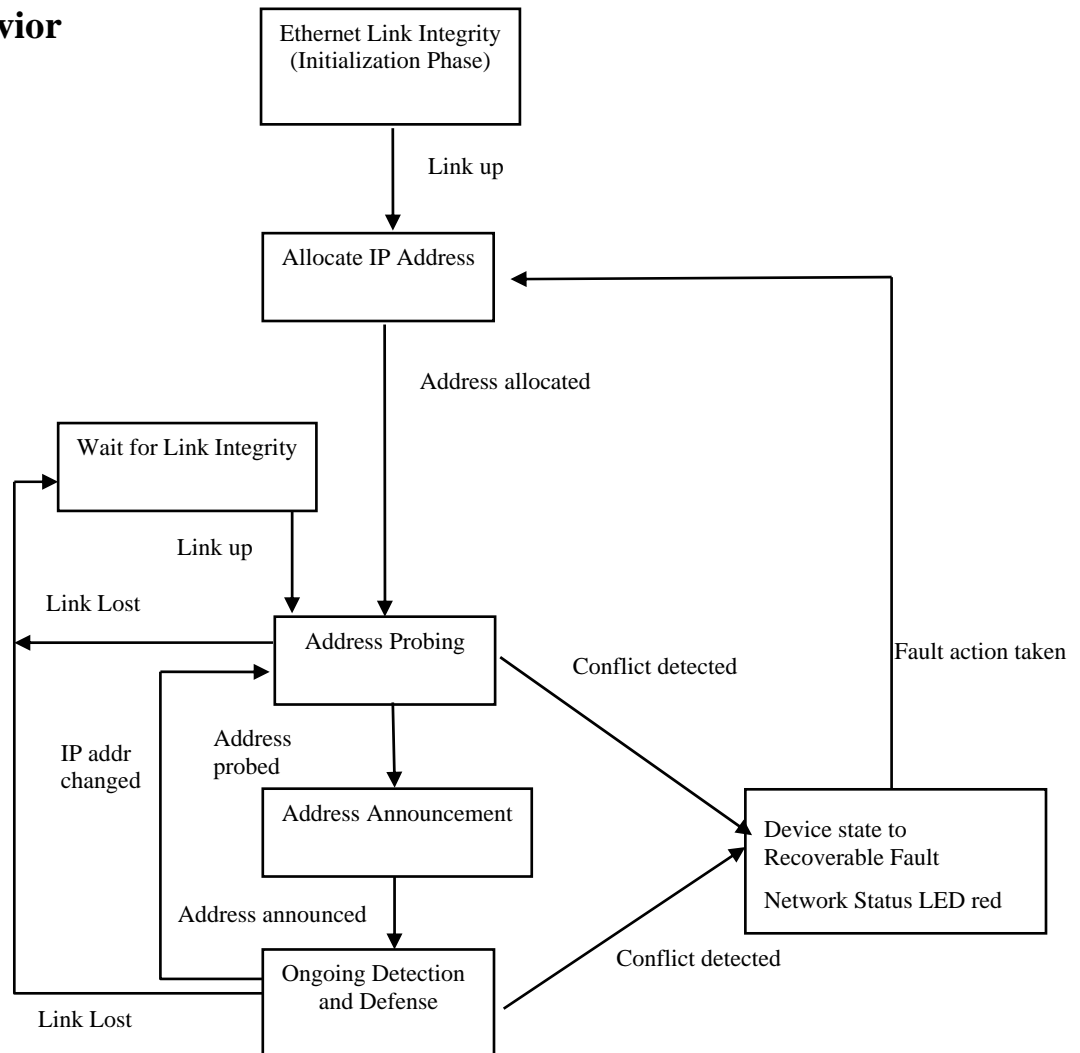
### 3.8 Summary of IPv4 ACD Behavior for EtherNet/IP Devices

As an aid to developers, the following table and diagrams summarize the IPv4 ACD behavior for EtherNet/IP devices:

Phase	Summary of Behavior	Causes of Conflict	Action Taken When Conflict is Detected
Initial Probing	<ol style="list-style-type: none"> <li>1. Don't probe until Ethernet Link Integrity detected.</li> <li>2. Delay from 0 to PROBE_WAIT, randomized.</li> <li>3. Issue PROBE_NUM ARP probes at interval between PROBE_MIN and PROBE_MAX.</li> <li>3. Listen for incoming ARP packets and check for conflict.</li> </ol>	<ol style="list-style-type: none"> <li>1. ARP probe with a matching Target IP Address.</li> <li>2. ARP message (request or reply) with a matching Sender IP Address.</li> </ol> <p>ARP packets with the Source Hardware address matching the devices' Ethernet MAC address are NOT conflicts.</p>	<ol style="list-style-type: none"> <li>1. Do not use IP address.</li> <li>2. Set device state to Recoverable Fault; set Network Status LED red.</li> <li>3. If address obtained via DHCP, device is RECOMMENDED to remain in conflict state.</li> <li>4. If address obtained via BOOTP, device must remain in conflict state.</li> <li>5. If the device has</li> </ol>

			<p>another communications port over which it can be configured, the device should allow the user to set a new, non-conflicting IP address.</p>
<p>Ongoing Detection</p>	<ol style="list-style-type: none"> <li>1. Listen for incoming ARP packets and check for conflict.</li> <li>2. Respond to ARP Probes (as well as normal ARP requests).</li> <li>3. The device should issue periodic ARP Probes (at intervals between ONGOING_PROBE_MIN and ONGOING_PROBE_MAX).</li> <li>4. Restart Initial Probing when Ethernet Link Integrity is lost and then regained.</li> </ol>	<ol style="list-style-type: none"> <li>1. ARP message (request or reply) with a matching Sender IP Address.</li> </ol> <p>ARP packets with the Source Hardware address matching the devices' Ethernet MAC address are NOT conflicts.</p>	<ol style="list-style-type: none"> <li>1. If no conflicts have been detected in the last seconds, the device should attempt to defend the address by sending an ARP announcement.</li> <li>2. If the conflict persists (i.e., another conflict is detected within DEFEND_INTERVAL), the device must follow the same conflict behavior as in the initial probing phase (see above).</li> </ol>

### Overall ACD Behavior



## 4 Implementation Considerations [Informational]

### 4.1 Summary

At present, the IPv4 ACD mechanism is not implemented in any common embedded operating systems. Consequently, developers must in general implement IPv4 ACD outside of the TCP/IP stack with which they have been provided.

This section discusses some of the considerations and issues of interest to developers when implementing the IPv4 ACD mechanism.

### 4.2 Implementation Requirements

In general, implementing IPv4 ACD in an EtherNet/IP device requires:

- Ability to send ARP probes and ARP announcements.
- Ability to receive ARP messages (requests and responses), check for conflicts and notify application code of the conflict.
- Ability to perform initial probes BEFORE initializing the Ethernet interface with the desired IP address.
- Ability to perform ongoing detection and defense and take the appropriate action if a conflict is detected (defend, release, etc.).
- Note that the ability to send and receive ARP messages will vary depending on embedded OS. In many cases, modification to the device's Ethernet driver is required.

### 4.3 Implementation Example

The IPv4 ACD mechanism has been implemented in several EtherNet/IP devices, using standard operating systems. The following summarizes the essential modifications required to implement the ACD mechanism:

- Create and send 'raw' ARP requests in order to perform the initial address probing, before initializing the device's IP address.
- Modifications to the Ethernet driver to receive ARP messages, check for a conflicting address, and signal that a conflict exists.
- Task-level code to process address conflicts (defend, release, change device status, etc.).

### 4.4 Example Pseudo-random Delay Algorithm

The following shows an example algorithm that might be used to produce an initial pseudo-random delay before sending out ARP probes, assuming an initial PROBE\_WAIT of 200 ms:

```
AcdProbeTime = 200;  
AcdDelay = EnetAddr[0];  
AcdDelay = ( AcdDelay << 1 ) ^ EnetAddr[1];  
AcdDelay = ( AcdDelay << 1 ) ^ EnetAddr[2];  
AcdDelay = ( AcdDelay << 1 ) ^ EnetAddr[3];
```

```
AcDelay = ( AcDelay << 1 ) ^ EnetAddr[4];  
AcDelay = ( AcDelay << 1 ) ^ EnetAddr[5];  
AcDelay = ( AcDelay - (( AcDelay mod AcProbeTime ) * AcProbeTime ));
```

#### **4.5 Issues With DHCP Clients**

Existing DHCP client implementations in commercial operating systems do not incorporate the IPv4 ACD mechanism. Unless access to DHCP client source code is possible, devices may need to perform the conflict detection mechanism outside of the DHCP client.

The preferred approach is to allow the DHCP client to manage obtaining the address lease but not control the actual setting of the IP address. The DHCP client obtains the address and lease, and the ACD implementation then probes for conflicts. If a conflict is found, the address is not used, the DHCP client is told to release the lease, and the device remains in the conflict state (as in section 3.2).

In some implementations, the DHCP client may use the obtained IP address before the application can perform the initial address probing. In this case, the application **MUST** perform the initial address probing as soon as is feasible. If a conflict is detected, the device must remain in the conflicted state (see section **Error! Reference source not found.**, Initial Address Probing).

#### **4.6 Future Direction**

Long term, it is hoped that embedded operating system vendors will adopt the IPv4 ACD mechanism.

## **5 Idiosyncrasies Observed with IPv4 ACD Behavior [Informational]**

### **5.1 Summary**

During testing of the initial IPv4 ACD implementation, a number of idiosyncrasies were observed. This section discusses observed IPv4 ACD behaviors that may be of interest to developers.

### **5.2 Infinite DHCP Client Loop**

Some DHCP clients can remain in an infinite loop when fixed addresses are used and an address conflict is detected. At least one embedded DHCP client implementation does its own rudimentary conflict detection by issuing an ARP probe of the newly acquired address. If a conflict is detected, the client requests a new address from the server. If the DHCP server responds with the same fixed IP address, the client enters into an endless loop of requesting an address from the server.

The ACD implementation may detect this condition by explicitly examining incoming ARP messages and knowing whether the DHCP client is running. In this situation, it is recommended that the device stop its DHCP client and remain in the address conflict state (see Section 3).

### **5.3 Switches Can Drop ARP probes**

When a switch port has the Spanning Tree Protocol, initial packets from the end devices will not be forwarded to the network for some period of time – typically 15-30 seconds. In this case, a device's initial ARP probes will not be seen by other devices. It is recommended that Spanning Tree be disabled on switch ports to which end devices are connected.

## 6 Test Cases [Informational]

### 6.1 Summary

This section suggests test cases for devices that implement the IPv4 ACD mechanism. The test cases are not meant to be an exhaustive list, but rather are a minimal set of suggested tests in order to verify an IPv4 ACD implementation.

### 6.2 Tests with Other IPv4 ACD Device

The following tests should be performed with the Device Under Test (DUT) and a second device that implements the IPv4 ACD mechanism. Tests should be performed with addresses assigned via BOOTP, DHCP (if supported), and static configuration.

Test	Device Under Test	Device 2 (with ACD)
1. Power up DUT and wait until it passes the initial probing phase. Power up device #2. No communications associations.	Keeps IP address	Detects conflict, gives up IP address.
2. Power up DUT. Establish communications with DUT. Power up device #2.	Keeps IP address. Communication with device remains open	Detects conflict, gives up IP address.
3. Power up device #2. Establish communications to device #2. Power up DUT with no cable. Attach cable to DUT.	Detects conflict, gives up IP address.	Keeps IP address. Communication with device remains open.
4. Power up DUT and wait until it passes the initial probing phase. Disconnect cable. Power up device #2 and wait until it passes the initial probing phase. Reattach cable to DUT.	Detects conflict, gives up IP address.	Keeps IP address
5. Power up DUT and device #2, with each connected to different hubs/switches with no link between the hubs/switches. Make sure each device has passed the initial probing phase. Then connect the hubs/switches.	Either DUT or device #2 detects conflict within ONGOING_PROBE_MAX, depending on which device probes first.  Also possible that both devices detect conflict, depending on timing of the ARP probes.	Either DUT or device #2 detects conflict within ONGOING_PROBE_MAX, depending on which device probes first.  Also possible that both devices detect conflict, depending on timing of the ARP probes.

<p>6. Power up DUT and device #2 “simultaneously”, such that they both are doing ARP probes at the same time.</p>	<p>Either DUT or device #2 detects conflict within ONGOING_PROBE_MAX, depending on which device probes first.</p> <p>Also possible that both devices detect conflict, depending on timing of the ARP probes.</p>	<p>Either DUT or device #2 detects conflict within ONGOING_PROBE_MAX, depending on which device probes first.</p> <p>Also possible that both devices detect conflict, depending on timing of the ARP probes.</p>
---	--	--

### 6.3 Tests with Devices Not Supporting IPv4 ACD

The following tests should be performed with the Device Under Test (DUT) and a second device that DOES NOT implement the IPv4 ACD mechanism. Tests should be performed with addresses assigned via BOOTP, DHCP (if supported), and static configuration.

Test	Device Under Test	Device 2 (without ACD)
1. Power up DUT and wait until it passes the initial probing phase. Power up device #2. No communication associations.	In general, detects conflict and gives up IP address. May defend address if other device issues ARP probes (e.g., Windows devices).	Variable
2. Power up DUT and wait until it passes the initial probing phase. Establish communications with DUT. Power up device #2.	Same as above.	Variable.
3. Power up device #2. Power up DUT with no cable. Attach cable to DUT.	Detects conflict, gives up IP address.	Variable.
4. Power up DUT and wait until it passes the initial probing phase. Disconnect cable. Power up device #2. Reattach cable to DUT.	Detects conflict, gives up IP address.	Variable.
5. Power up DUT and device #2, with each connected to different hubs/switches with no link between the hubs/switches. Make sure DUT has passed the initial probing phase. Then connect the hubs/switches.	Conflict should be detected within ONGOING_PROBE_MAX, gives up IP address.	Variable.

## 7 References

- [Cheshire05] Stuart Cheshire, "IPv4 Address Conflict Detection", Internet Draft, 11-July-2005.



The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

## Abstract

When two hosts on the same link attempt to use the same IPv4 address at the same time (except in rare special cases where this has been arranged by prior coordination) problems ensue for one or both hosts.

This document describes (i) a simple precaution that a host can take in advance to help prevent this misconfiguration from happening, and (ii) if this misconfiguration does occur, a simple mechanism by which

a host can passively detect after-the-fact that it has happened, so that the host or administrator may respond to rectify the problem.

## 1. Introduction

Historically, accidentally configuring two Internet hosts with the same IP address has often been an annoying and hard-to-diagnose problem.

This is unfortunate, because the existing ARP protocol provides an easy way for a host to detect this kind of misconfiguration and report it to the user. The DHCP specification [RFC2131] briefly mentions the role of ARP in detecting misconfiguration, as illustrated in the following three excerpts from RFC 2131:

- o the client SHOULD probe the newly received address, e.g., with ARP.
- o The client SHOULD perform a final check on the parameters (e.g., ARP for allocated network address)
- o If the client detects that the address is already in use (e.g., through the use of ARP), the client MUST send

a DHCPDECLINE message to the server

Unfortunately, the DHCP specification does not give any guidance to implementers concerning the number of ARP packets to send, the interval between packets, the total time to wait before concluding that an address may safely be used, or indeed even which kinds of packets a host should be listening for, in order to make this determination. It leaves unspecified the action a host should take if, after concluding that an address may safely be used, it subsequently discovers that it was wrong. It also fails to specify what precautions a DHCP client should take to guard against pathological failure cases, such as DHCP server that repeatedly OFFERs the same address, even though it has been DECLINED multiple times.

The authors of the DHCP specification may have have been justified in thinking at the time that the answers to these questions seemed too simple, obvious and straightforward to be worth mentioning, but unfortunately this left some of the burden of protocol design to each individual implementer. This document seeks to remedy this omission by clearly specifying the required actions for:

1. Determining whether use of an address is likely to lead to an addressing conflict. This includes (a) the case where the address is already actively in use by another host on the same link, and (b) the case where two hosts are inadvertently about to begin using the same address, and both are simultaneously in the process of probing to determine whether the address may safely be used. (Section 2.1.)

2. Subsequent passive detection that another host on the network is inadvertently using the same address. Even if all hosts observe precautions to avoid using an address that is already in use, conflicts can still occur if two hosts are out of communication at

the time of initial interface configuration. This could occur with wireless network interfaces if the hosts are temporarily out of range, or with Ethernet interfaces if the link between two Ethernet hubs is not functioning at the time of address configuration. A well-designed host will handle not only conflicts detected during interface configuration, but also conflicts detected later, for the entire duration of the time that the host is using the address. (Section 2.4.)

3. Rate-limiting of address acquisition attempts in the case of an excessive number of repeated conflicts. (Section 2.1.)

The utility of IPv4 Address Conflict Detection (ADC) is not limited to DHCP clients. No matter how an address was configured, whether via manual entry by a human user, via information received from a DHCP server, or via any other source of configuration information, detecting conflicts is useful. Upon detecting a conflict, the configuring agent should be notified of the error. In the case where

the configuring agent is a human user, that notification may take the

form of an error message on a screen, an SNMP trap, or an error message sent via pager. In the case of a DHCP server, that notification takes the form of a DHCP DECLINE message sent to the server. In the case of configuration by some other kind of software,

that notification takes the form of an error indication to the software in question, to inform it that the address it selected is in conflict with some other host on the network. The configuring software may choose to cease network operation, or it may automatically select a new address so that the host may re-establish IP connectivity as soon as possible.

Allocation of IPv4 Link-Local Addresses [RFC3927] can be thought of as a special-case of this mechanism, where the configuring agent is a pseudo-random number generator, and the action it takes upon being notified of a conflict is to pick a different random number and try again. In fact, this is exactly how IPv4 Link-Local Addressing was

implemented in Mac OS 9 back in 1998. If the DHCP client failed to get a response from any DHCP server, it would simply make up a fake response containing a random 169.254.x.x address. If the ARP module reported a conflict for that address, then the DHCP client would try again, making up a new random 169.254.x.x address as many times as was necessary until it succeeded. Implementing ACD as a standard feature of the networking stack has the side-effect that it means that half the work for IPv4 Link-Local Addressing is already done.

### 1.1. Conventions and Terminology Used in this Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

Wherever this document uses the term "sender IP address" or "target IP address" in the context of an ARP packet, it is referring to the fields of the ARP packet identified in the ARP specification [RFC826]

as "ar\$spa" (Sender Protocol Address) and "ar\$tpa" (Target Protocol Address) respectively. For the usage of ARP described in this document, each of these fields always contains an IPv4 address.

In this document, the term "ARP Probe" is used to refer to an ARP Request packet, broadcast on the local link, with an all-zero 'sender

IP address'. The 'sender hardware address' MUST contain the hardware

address of the interface sending the packet. The 'sender IP address'

field MUST be set to all zeroes, to avoid polluting ARP caches in other hosts on the same link in the case where the address turns out to be already in use by another host. The 'target hardware address' field is ignored and SHOULD be set to all zeroes. The 'target IP address' field MUST be set to the address being probed. An "ARP Probe" conveys both a question ("Is anyone using this address?") and an implied statement ("This is the address I intend to use.").

In this document, the term "ARP Announcement" is used to refer to an ARP Request packet, broadcast on the local link, identical to the ARP probe described above, except that both the sender and target IP address fields contain the IP address being announced. It conveys a stronger statement than an "ARP Probe", namely, "This is the address I am now using."

The following timing constants are used in this protocol; they are not intended to be user-configurable. These constants are referenced in Section 2, which describes the operation of the protocol in detail.

PROBE_WAIT	1 second	(initial random delay)
PROBE_NUM	3	(number of probe packets)
PROBE_MIN	1 second	(minimum delay until repeated probe)
PROBE_MAX	2 seconds	(maximum delay until repeated probe)
ANNOUNCE_WAIT	2 seconds	(delay before announcing)
ANNOUNCE_NUM	2	(number of announcement packets)
ANNOUNCE_INTERVAL	2 seconds	(time between announcement packets)
MAX_CONFLICTS	10	(max conflicts before rate limiting)
RATE_LIMIT_INTERVAL	60 seconds	(delay between successive attempts)
DEFEND_INTERVAL	10 seconds	(minimum interval between defensive ARPs).

## 1.2 Relationship to RFC 826

This document does not modify any of the protocol rules in RFC 826. It does not modify the packet format, or the meaning of any of the fields. The existing rules for "Packet Generation" and "Packet Reception" still apply exactly as specified in RFC 826.

This document expands on RFC 826 by specifying:

- (1) that a specific ARP Request should be generated when an interface is configured, to discover if the address is already in use, and

(2) an additional trivial test that should be performed on each received ARP packet, to facilitate passive ongoing conflict detection. This additional test creates no additional packet overhead on the network (no additional packets are sent) and negligible additional CPU burden on hosts, since every host implementing ARP is *\*already\** required to process every received ARP packet according to the "Packet Reception" rules specified in RFC 826. These rules already include checking to see if the sender IP address of the ARP packet appears in any of the entries in the host's ARP cache; the additional test is simply to check to see if the sender IP address is the host's *\*own\** IP address, potentially as little as a single additional machine instruction on many architectures.

As already specified in RFC 826, an ARP Request packet serves two functions, an assertion and a question:

*\* Assertion:*

The fields "ar\$sha" (Sender Hardware Address) and "ar\$spa" (Sender Protocol Address) together serve as an assertion of a fact, that the stated Protocol Address is mapped to the stated Hardware Address.

*\* Question:*

The fields "ar\$tha" (Target Hardware Address, zero) and "ar\$tpa" (Target Protocol Address) serve as a question, asking, for the stated Protocol Address, to which Hardware Address it is mapped.

This document clarifies what it means to have one without the other.

### 1.2.1 Broadcast Replies

In some applications of IPv4 Address Conflict Detection (ACD), it

may be advantageous to deliver ARP Replies using broadcast instead of unicast because this allows address conflicts to be detected sooner than might otherwise happen. For example, "Dynamic Configuration of IPv4 Link-Local Addresses" [RFC3927] uses ACD exactly as specified here, but additionally specifies that ARP Replies should be sent using broadcast, because in that context the trade-off of increased broadcast traffic in exchange for improved reliability and fault-tolerance was deemed to be an appropriate one. There may be other future specifications where the same trade-off is appropriate.

RFC 826 implies that replies to ARP Requests are usually delivered using unicast, but it is also acceptable to deliver ARP Replies using broadcast. The "Packet Reception" rules in RFC 826 specify that the content of the "ar\$spa" field should be processed *before* examining the "ar\$op" field, so any host that correctly implements the Packet Reception algorithm specified in RFC 826 will correctly handle ARP Replies delivered via link-layer broadcast.

### 1.3. Applicability

This specification applies to all IEEE 802 Local Area Networks (LANs)

[802], including Ethernet [802.3], Token-Ring [802.5] and IEEE 802.11

wireless LANs [802.11], as well as to other link-layer technologies that operate at data rates of at least 1 Mbps, have a round-trip latency of at most one second, and use ARP [RFC826] to map from IP addresses to link-layer hardware addresses. Wherever this document uses the term "IEEE 802", the text applies equally to any of these network technologies.

Link-layer technologies that support ARP but operate at rates below 1 Mbps or latencies above one second will still work correctly with this protocol, but more often may have to handle late conflicts detected after the Probing phase has completed. On these kinds

of link, it may be desirable to specify different values for the following parameters:

- (a) PROBE\_NUM, PROBE\_MIN, and PROBE\_MAX, the number of, and interval between, ARP probes, explained in Section 2.1.1.
- (b) ANNOUNCE\_NUM and ANNOUNCE\_INTERVAL, the number of, and interval between, ARP announcements, explained in Section 2.3.
- (c) RATE\_LIMIT\_INTERVAL and MAX\_CONFLICTS, controlling the maximum rate at which address claiming may be attempted, explained in Section 2.1.1.

(d) DEFEND\_INTERVAL, the time interval between conflicting ARPs below which a host MUST NOT attempt to defend its address, explained in Section 2.4.

Link-layer technologies that do not support ARP may be able to use other techniques for determining whether a particular IP address is currently in use. However, implementing Address Conflict Detection for such networks is outside the scope of this document.

For the protocol specified in this document to be effective, it is not necessary that every host on the link implements it. For a given host implementing this specification to be protected against accidental address conflicts, all that is required is that the peers on the same link correctly implement the ARP protocol as given in RFC 826. To be specific, when a peer host receives an ARP Request where the Target Protocol Address of the ARP Request matches (one of) that host's IP address(es) configured on that interface, then as long as it properly responds with a correctly-formatted ARP Reply, the querying host will be able to detect that the address is already in use.

The specifications in this document allow hosts to detect conflicts between two hosts using the same address on the same physical link.

ACD does not detect conflicts between two hosts using the same address on different physical links, and indeed it should not. For example, the address 10.0.0.1 [RFC1918] is in use by countless devices on countless private networks throughout the world, and this is not a conflict, because they are on different links. It would only be a conflict if two such devices were to be connected to the same link, and when this happens (as it sometimes does), this is a perfect example of a situation where ACD is extremely useful to detect and report (and/or automatically correct) this error.

For the purposes of this document, a set of hosts is considered to be "on the same link" if:

- when any host A from that set sends a packet to any other host B in that set, using unicast, multicast, or broadcast, the entire link-layer packet payload arrives unmodified, and
- a broadcast sent over that link by any host from that set of hosts can be received by every other host in that set

The link-layer *\*header\** may be modified, such as in Token Ring Source

Routing [802.5], but not the link-layer *\*payload\**. In particular, if

any device forwarding a packet modifies any part of the IP header or IP payload then the packet is no longer considered to be on the same link. This means that the packet may pass through devices such as repeaters, bridges, hubs or switches and still be considered to be on

the same link for the purpose of this document, but not through a device such as an IP router that decrements the TTL or otherwise modifies the IP header.

## 2. Address Probing, Announcing, Conflict Detection and Defense

This section describes initial probing to safely determine whether an

address is already in use, announcing the chosen address, ongoing

conflict checking, and optional use of broadcast ARP Replies to provide faster conflict detection.

## 2.1 Probing an Address

Before beginning to use an IPv4 address (whether received from manual configuration, DHCP, or some other means), a host implementing this specification MUST test to see if the address is already in use, by broadcasting ARP Probe packets. This also applies when when a network interface transitions from an inactive to an active state, when a computer awakes from sleep, when a link-state change signals that an Ethernet cable has been connected, when an 802.11 wireless interface associates with a new base station, or any other change in connectivity where a host becomes actively connected to a logical link.

A host MUST NOT perform this check periodically as a matter of course. This would be a waste of network bandwidth, and is unnecessary due to the ability of hosts to passively discover conflicts, as described in Section 2.4.

### 2.1.1.1. Probe Details

A host probes to see if an address is already in use by broadcasting an ARP Request for the desired address. The client MUST fill in the 'sender hardware address' field of the ARP Request with the hardware address of the interface through which it is sending the packet.

The

'sender IP address' field MUST be set to all zeroes, to avoid polluting ARP caches in other hosts on the same link in the case where the address turns out to be already in use by another host.

The 'target hardware address' field is ignored and SHOULD be set to all zeroes. The 'target IP address' field MUST be set to the address

being probed. An ARP Request constructed this way with an all-zero 'sender IP address' is referred to as an "ARP Probe".

When ready to begin probing, the host should then wait for a random time interval selected uniformly in the range zero to PROBE\_WAIT seconds, and should then send PROBE\_NUM probe packets, each of these probe packets spaced randomly, PROBE\_MIN to PROBE\_MAX seconds apart. This initial random delay helps ensure that a large number of hosts powered on at the same time do not all send their initial probe packets simultaneously.

If during this period, from the beginning of the probing process until ANNOUNCE\_WAIT seconds after the last probe packet is sent, the host receives any ARP packet (Request \*or\* Reply) on the interface where the probe is being performed where the packet's 'sender IP address' is the address being probed for, then the host MUST treat this address as being in use by some other host, and should indicate to the configuring agent (human operator, DHCP server, etc.) that the proposed address is not acceptable.

In addition, if during this period the host receives any ARP Probe where the packet's 'target IP address' is the address being probed for, and the packet's 'sender hardware address' is not the hardware address of the interface the host is attempting to configure, then the host MUST similarly treat this as an address conflict and signal an error to the configuring agent as above. This can occur if two (or more) hosts have, for whatever reason, been inadvertently configured with the same address, and both are simultaneously in the process of probing that address to see if it can safely be used.

NOTE: The check that the packet's 'sender hardware address' is not the hardware address of any of the host's interfaces is important. Some kinds of Ethernet hub (often called a "buffered repeater") and many wireless access points may "rebroadcast" any received broadcast packets to all recipients, including the original sender itself. For this reason, the precaution described above is necessary to ensure that a host is not confused when it sees its own ARP packets echoed back.

A host implementing this specification MUST take precautions to limit

the rate at which it probes for new candidate addresses: If the host experiences MAX\_CONFLICTS or more address conflicts on a given interface, then the host MUST limit the rate at which it probes for new addresses on this interface to no more than one per RATE\_LIMIT\_INTERVAL. This is to prevent catastrophic ARP storms in pathological failure cases, such as a defective DHCP server that repeatedly assigns the same address to every host that asks for one. This rate limiting rule applies not only to conflicts experienced during the initial probing phase, but also to conflicts experienced later, as described in Section 2.4 "Ongoing Address Conflict Detection and Address Defense".

If, by ANNOUNCE\_WAIT seconds after the transmission of the last ARP Probe no conflicting ARP Reply or ARP Probe has been received, then the host has successfully determined that the desired address may be used safely.

## 2.2 Shorter Timeouts on Appropriate Network Technologies

Network technologies may emerge for which shorter delays are appropriate than those required by this document. A subsequent IETF publication may be produced providing guidelines for different values

for PROBE\_WAIT, PROBE\_NUM, PROBE\_MIN and PROBE\_MAX on those technologies.

If the situation arises where different hosts on a link are using different timing parameters, this does not cause any problems. This protocol is not dependent on all hosts on a link implementing the same version of the protocol; indeed, this protocol is not dependent on all hosts on a link implementing the protocol at all. All that is

required is that all hosts implement ARP as specified in RFC 826, and

correctly answer ARP Requests they receive. In the situation where different hosts are using different timing parameters, all that will

happen is that some hosts will configure their interfaces quicker than others. In the unlikely event that an address conflict is not detected during the address probing phase, the conflict will still be detected by the Ongoing Address Conflict Detection described below in Section 2.4.

### 2.3 Announcing an Address

Having probed to determine that a desired address may be used safely, a host implementing this specification MUST then announce that it is commencing to use this address by broadcasting ANNOUNCE\_NUM ARP announcements, spaced ANNOUNCE\_INTERVAL seconds apart. An ARP announcement is identical to the ARP Probe described above, except that now the sender and target IP addresses are both set to the host's newly selected IPv4 address. The purpose of these ARP announcements is to make sure that other hosts on the link do not have stale ARP cache entries left over from some other host that may previously have been using the same address. The host may begin legitimately using the IP address immediately after sending the first of the two ARP announcements, and the sending of the second ARP announcement may be completed asynchronously, concurrent with other networking operations the host may wish to perform.

### 2.4 Ongoing Address Conflict Detection and Address Defense

Address conflict detection is not limited to only the time of initial interface configuration, when a host is sending ARP probes. Address conflict detection is an ongoing process that is in effect for as long as a host is using an address. At any time, if a host receives an ARP packet (Request \*or\* Reply) where the 'sender IP address' is (one of) the host's own IP address(es) configured on that interface, but the 'sender hardware address' does not match any of the host's own interface addresses, then this is a conflicting ARP packet, indicating some other host also thinks it is validly using this

address. To resolve the address conflict, a host MUST respond to a conflicting ARP packet as described in either (a), (b) or (c) below:

(a) Upon receiving a conflicting ARP packet, a host MAY elect to immediately cease using the address, and signal an error to the configuring agent as described above, or

(b) If a host currently has active TCP connections or other reasons to prefer to keep the same IPv4 address, and it has not seen any other conflicting ARP packets within the last DEFEND\_INTERVAL seconds, then it MAY elect to attempt to defend its address by recording the time that the conflicting ARP packet was received, and then broadcasting one single ARP announcement, giving its own IP and hardware addresses as the sender addresses of the ARP. Having done this, the host can then continue to use the address normally without any further special action. However, if this is not the first conflicting ARP packet the host has seen, and the time recorded for the previous conflicting ARP packet is recent, within DEFEND\_INTERVAL

seconds, then the host MUST immediately cease using this address and signal an error to the configuring agent as described above. This is necessary to ensure that two hosts do not get stuck in an endless loop with both hosts trying to defend the same address.

(c) If a host has been configured such that it should not give up its address under any circumstances (perhaps because it is the kind of device that needs to have a well-known stable IP address, such as a link's default router, or a DNS server) then it MAY elect to defend its address indefinitely. If such a host receives a conflicting ARP packet, then it should take appropriate steps to log useful information such as source Ethernet address from the ARP packet, and inform an administrator of the problem. The number of such notifications should be appropriately controlled to prevent an excessive number of error reports being generated. If the host has not seen any other conflicting ARP packets recently within the last DEFEND\_INTERVAL seconds then it MUST record the time that the

conflicting ARP packet was received, and then broadcast one single ARP announcement, giving its own IP and hardware addresses. Having done this, the host can then continue to use the address normally without any further special action. However, if this is not the first conflicting ARP packet the host has seen, and the time recorded

for the previous conflicting ARP packet is within DEFEND\_INTERVAL seconds then the host MUST NOT send another defensive ARP announcement. This is necessary to ensure that two misconfigured hosts do not get stuck in an endless loop flooding the network with broadcast traffic while they both try to defend the same address.

A host wishing to provide reliable network operation MUST respond to conflicting ARP packets as described in (a), (b) or (c) above. Ignoring conflicting ARP packets results in seemingly random network failures which can be hard to diagnose and very frustrating for human users.

Forced address reconfiguration may be disruptive, causing TCP connections to be broken. However, such disruptions should be exceedingly rare, and if inadvertent address duplication happens, then disruption of communication is inevitable. It is not possible for two different hosts using the same IP address on the same network to operate reliably.

Before abandoning an address due to a conflict, hosts SHOULD actively attempt to reset any existing connections using that address. This mitigates some security threats posed by address reconfiguration, as discussed in Section 3.

For most client machines that do not need a fixed IP address, immediately requesting the configuring agent (human user, DHCP client, etc.) to configure a new address as soon as the conflict is

detected is the best way to restore useful communication as quickly as possible. The mechanism described above of broadcasting a single ARP announcement to defend the address mitigates the problem somewhat, by helping to improve the chance that one of the two conflicting hosts may be able to retain its address.

## 2.5 Broadcast ARP Replies

In a carefully-run network with manually-assigned addresses, or a network with a reliable DHCP server and reliable DHCP clients, address conflicts should occur only in rare failure scenarios, so the passive monitoring described above in Section 2.4 is adequate.

If two hosts are using the same IP address, then sooner or later one or other host will broadcast an ARP Request, which the other will see, allowing the conflict to be detected and consequently resolved.

It is possible however, that a conflicting configuration may persist for a short time before it is detected. Suppose that two hosts A and

B have been inadvertently assigned the same IP address X. Suppose further that at the time they were both probing to determine whether the address could safely be used, the communication link between them

was non-functional for some reason, so neither detected the conflict at interface-configuration time. Suppose now that the communication link is restored, and a third host C broadcasts an ARP Request for address X. Unaware of any conflict, both hosts A and B will send unicast ARP Replies to host C. Host C will see both Replies, and may

be a little confused, but neither host A nor B will see the other's Reply, and neither will immediately detect that there is a conflict to be resolved. Hosts A and B will continue to be unaware of the conflict until one or other broadcasts an ARP Request of their own.

If quicker conflict detection is desired, this may be achieved by having hosts send ARP Replies using link-level broadcast, instead of sending only ARP Requests via broadcast, and Replies via unicast.

This is NOT RECOMMENDED for general use, but other specifications building on IPv4 ACD may choose to specify broadcast ARP Replies if appropriate. For example, "Dynamic Configuration of IPv4 Link-Local Addresses" [RFC3927] specifies broadcast ARP Replies because in that context, detection of address conflicts using IPv4 ACD is not merely a backup precaution to detect failures of some other configuration mechanism; detection of address conflicts using IPv4 ACD is the sole configuration mechanism.

Sending ARP Replies using broadcast does increase broadcast traffic, but in the worst case by no more than a factor of two. In the traditional usage of ARP, a unicast ARP Reply only occurs in response

to a broadcast ARP Request, so sending these via broadcast instead means that we generate at most one broadcast Reply in response to each existing broadcast Request. On many networks, ARP traffic is such an insignificant proportion of the total traffic that doubling it makes no practical difference. However, this may not be true of all networks, so broadcast ARP Replies SHOULD NOT be used universally. Broadcast ARP Replies should be used where the benefit of faster conflict detection outweighs the cost of increased broadcast traffic and increased packet processing load on the participant network hosts.

### 3. Security Considerations

IPv4 Address Conflict Detection (ACD) is based on ARP [RFC826] and inherits the security vulnerabilities of this protocol. A malicious host may send fraudulent ARP packets on the network, interfering with

the correct operation of other hosts. For example, it is easy for a host to answer all ARP Requests with Replies giving its own hardware address, thereby claiming ownership of every address on the network.

This specification makes this existing ARP vulnerability no worse, and in some ways makes it better: Instead of failing silently with no indication why, hosts implementing this specification either attempt

to reconfigure automatically, or at least inform the human user of what is happening.

If a host willingly selects a new address in response to an ARP conflict, as described in Section 2.4 subsection (a), this potentially makes it easier for malicious attackers on the same link to hijack TCP connections. Having a host actively reset any existing connections before abandoning an address helps mitigate this risk.

#### 4. Historical Note

A widely-known, but ineffective, duplicate address detection technique called "Gratuitous ARP" is found in many deployed systems [Ste94]. What Stevens describes as Gratuitous ARP is the exact same packet that this document refers to by the more descriptive term "ARP

Announcement". This traditional Gratuitous ARP implementation sends only a single ARP Announcement when an interface is first configured.

The result is that the victim (the existing address holder) logs an error, and the offender continues operation, often without even detecting any problem. Both machines then typically proceed to try to use the same IP address, and fail to operate properly because they

are each constantly resetting the other's TCP connections. The human

administrator is expected to notice the log message on the victim machine and repair the damage after the fact. Typically this has to be done by physically going to the machines in question, since in this state neither is able to keep a TCP connection open for long enough to do anything useful over the network.

The problems caused by this ineffective duplicate address detection technique are illustrated by the fact that (as of August 2004) the top Google search results for the phrase "Gratuitous ARP" are articles describing how to disable it.

However, implementers of IPv4 Address Conflict Detection should be

aware that, as of this writing, Gratuitous ARP is still widely deployed. The steps described in Sections 2.1 and 2.4 of this document help make a host robust against misconfiguration and address conflicts, even when the other host is *\*not\** playing by the same rules.

5. Why are ARP Announcements performed using ARP Request packets and not ARP Reply packets?

During IETF deliberation of IPv4 Address Conflict Detection from 2000 to 2005, a question that kept being asked repeatedly was, "Shouldn't ARP Announcements be performed using gratuitous ARP Reply packets?"

On the face of it, this seems reasonable. A conventional ARP Reply is an answer to a question. If in fact no question had been asked, then it would be reasonable to describe such a reply as gratuitous. This description would seem to apply perfectly to an ARP Announcement: an answer to an implied question that in fact no one asked.

However reasonable this may seem in principle, there are two reasons why in practice ARP Request packets are the better choice. One is historical precedent, and the other is practicality.

The historical precedent is that, as described above in Section 4, Gratuitous ARP is described in Stevens Networking [Ste94] as using ARP Request packets. BSD Unix, Windows, Mac OS 9, Mac OS X, etc. all use ARP Request packets as described in Stevens. At this stage, trying to mandate that they all switch to using ARP Reply packets would be futile.

The practical reason is that ARP Request packets are more likely to work correctly with more existing ARP implementations, some of which may not implement RFC 826 correctly. The Packet Reception rules in RFC 826 state that the opcode is the last thing to check in packet processing, so it really shouldn't matter, but there may be "creative" implementations that have different packet processing

depending on the ar\$op field, and there are several reasons why these

are more likely to accept gratuitous ARP Requests than gratuitous ARP

Replies:

\* An incorrect ARP implementation may expect that ARP Replies are only sent via unicast. RFC 826 does not say this, but an incorrect

implementation may assume it, and the "principle of least surprise"

dictates that where there are two or more ways to solve a networking problem that are otherwise equally good, the one with the fewest unusual properties is the one likely to have the fewest interoperability problems with existing implementations. An ARP Announcement needs to broadcast information to all hosts on the link. Since ARP Request packets are always broadcast, and ARP Reply packets are not, receiving an ARP Request packet via broadcast is less surprising than receiving an ARP Reply packet via

broadcast.

\* An incorrect ARP implementation may expect that ARP Replies are only received in response to ARP Requests that have been issued recently by that implementation. Unexpected unsolicited Replies may be ignored.

\* An incorrect ARP implementation may ignore ARP Replies where ar\$tha doesn't match its hardware address.

\* An incorrect ARP implementation may ignore ARP Replies where ar\$tpa doesn't match its IP address.

In summary, there are more ways that an incorrect ARP implementation might plausibly reject an ARP Reply (which usually occurs as a result

of being solicited by the client) than an ARP Request (which is already expected to occur unsolicited).

## 6. IANA Considerations

This specification does not request the creation of any new parameter

registries, nor does it require any other IANA assignments.

## 7. Acknowledgments

This document arose as a result of discussions on link-local addressing, where it was not clear to many readers which elements of link-local address management were specific to that particular problem, and which elements were generic and applicable to all IPv4 address configuration mechanisms. The following people made valuable

comments in the course of that work and/or the subsequent editing of this document: Bernard Aboba, Randy Bush, Jim Busse, James Carlson, Alan Cox, Pavani Diwanji, Ralph Droms, Donald Eastlake 3rd, Alex Elder, Peter Ford, Spencer Giacalone, Josh Graessley, Erik Guttman, Myron Hattig, Hugh Holbrook, Richard Johnson, Kim Yong-Woon,

Marc Krochmal, Rod Lopez, Satish Mundra, Thomas Narten, Erik Nordmark, Howard Ridenour, Pekka Savola, Daniel Senie, Dieter Siegmund, Valery Smyslov and Ryan Troll.

## 8. Copyright Notice

Copyright (C) The Internet Society (2005).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights. For the purposes of this document, the term "BCP 78" refers exclusively to RFC 3978, "IETF Rights in Contributions", published March 2005.

This document and the information contained herein are provided on an

"AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS

OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET

ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## 9. Normative References

- [RFC826] D. Plummer, "An Ethernet Address Resolution Protocol -or- Converting Network Addresses to 48-bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, November 1982.
- [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

## 10. Informative References

- [802] IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture, ANSI/IEEE Std 802, 1990.
- [802.3] ISO/IEC 8802-3 Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Common specifications - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, (also ANSI/IEEE Std 802.3-1996), 1996.
- [802.5] ISO/IEC 8802-5 Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Common specifications - Part 5: Token ring access method and physical layer specifications, (also ANSI/IEEE Std 802.5-1998), 1998.

- [802.11] Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11-1999, 1999.
- [RFC2131] R. Droms, "Dynamic Host Configuration Protocol", RFC 2131, March 1997.
- [RFC3927] S. Cheshire, B. Aboba, E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, May 2005.
- [Ste94] W. Stevens, "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Wesley, 1994.

#### 11. Author's Address

Stuart Cheshire  
Apple Computer, Inc.  
1 Infinite Loop  
Cupertino  
California 95014  
USA

Phone: +1 408 974 3207  
EMail: rfc@stuartcheshire.org