Network Working Group
Request for Comments: 5227
Updates: 826
Category: Standards Track

S. Cheshire
Apple Inc.
July 2008

IPv4 Address Conflict Detection

Status of This Memo

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.

Table of Contents

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 Address Resolution Protocol (ARP) 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:

- the client SHOULD probe the newly received address, e.g., with ARP

- The client SHOULD perform a final check on the parameters (e.g., ARP for allocated network address)

- 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 a DHCP server that repeatedly OFFERs the same address, even though it has been DECLINEd multiple times.

The authors of the DHCP specification may 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 (ACD) 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, a Simple Network Management Protocol (SNMP) notification, or an error message sent via text message to a mobile phone. 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 hope 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 used in this protocol are referenced in Section 2, which describes the operation of the protocol in detail. (Note that the values listed here are fixed constants; they are not intended to be modifiable by implementers, operators, or end users. These constants are given symbolic names here to facilitate the writing of future standards that may want to reference this document with different values for these named constants; however, at the present time no such future standards exist.)

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

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.