Network Working Group
Request for Comments: 2390
Obsoletes: 1293
Category: Standards Track

T. Bradley
Avici Systems, Inc.
C. Brown
Consultant
A. Malis
Ascend Communications, Inc.

September 1998

# Inverse Address Resolution Protocol

## Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

## Copyright Notice



## 2. Abstract

This memo describes additions to ARP that will allow a station to request a protocol address corresponding to a given hardware address. Specifically, this applies to Frame Relay stations that may have a Data Link Connection Identifier (DLCI), the Frame Relay equivalent of a hardware address, associated with an established Permanent Virtual Circuit (PVC), but do not know the protocol address of the station on the other side of this connection. It will also apply to other networks with similar circumstances.

This memo replaces RFC 1293. The changes from RFC 1293 are minor changes to formalize the language, the additions of a packet diagram and an example in section 7.2, and a new security section.

## 3. Conventions

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [5].

## 4. Introduction

This document will rely heavily on Frame Relay as an example of how the Inverse Address Resolution Protocol (InARP) can be useful. It is not, however, intended that InARP be used exclusively with Frame Relay. InARP may be used in any network that provides destination hardware addresses without indicating corresponding protocol addresses.

## 5. Motivation

The motivation for the development of Inverse ARP is a result of the desire to make dynamic address resolution within Frame Relay both possible and efficient. Permanent virtual circuits (PVCs) and eventually switched virtual circuits (SVCs) are identified by a Data Link Connection Identifier (DLCI). These DLCIs define a single virtual connection through the wide area network (WAN) and may be thought of as the Frame Relay equivalent to a hardware address. Periodically, through the exchange of signaling messages, a network may announce a new virtual circuit with its corresponding DLCI. Unfortunately, protocol addressing is not included in the announcement. The station receiving such an indication will learn of the new connection, but will not be able to address the other side. Without a new configuration or a mechanism for discovering the protocol address of the other side, this new virtual circuit is unusable.

Other resolution methods were considered to solve the problems, but were rejected. Reverse ARP [4], for example, seemed like a good candidate, but the response to a request is the protocol address of the requesting station, not the station receiving the request. IP specific mechanisms were limiting since they would not allow resolution of other protocols other than IP. For this reason, the ARP protocol was expanded.

Inverse Address Resolution Protocol (InARP) will allow a Frame Relay station to discover the protocol address of a station associated with the virtual circuit. It is more efficient than sending ARP messages on every VC for every address the system wants to resolve and it is more flexible than relying on static configuration.

## 6. Packet Format

Inverse ARP is an extension of the existing ARP. Therefore, it has the same format as standard ARP.

| | | |
|---|---|---|
| ar$hrd | 16 bits | Hardware type |
| ar$pro | 16 bits | Protocol type |
| ar$hln | 8 bits | Byte length of each hardware address (n) |
| ar$pln | 8 bits | Byte length of each protocol address (m) |
| ar$op | 16 bits | Operation code |
| ar$sha | nbytes | source hardware address |
| ar$spa | mbytes | source protocol address |
| ar$tha | nbytes | target hardware address |
| ar$tpa | mbytes | target protocol address |

Possible values for hardware and protocol types are the same as those for ARP and may be found in the current Assigned Numbers RFC [2].

Length of the hardware and protocol address are dependent on the environment in which InARP is running. For example, if IP is running over Frame Relay, the hardware address length is either 2, 3, or 4, and the protocol address length is 4.

The operation code indicates the type of message, request or response.

```
InARP request  = 8
InARP response = 9
```

These values were chosen so as not to conflict with other ARP extensions.

## 7. Protocol Operation

Basic InARP operates essentially the same as ARP with the exception that InARP does not broadcast requests. This is because the hardware address of the destination station is already known.

When an interface supporting InARP becomes active, it should initiate the InARP protocol and format InARP requests for each active PVC for which InARP is active. To do this, a requesting station simply formats a request by inserting its source hardware, source protocol addresses and the known target hardware address. It then zero fills the target protocol address field. Finally, it will encapsulate the packet for the specific network and send it directly to the target station.

Upon receiving an InARP request, a station may put the requester's protocol address/hardware address mapping into its ARP cache as it would any ARP request. Unlike other ARP requests, however, the receiving station may assume that any InARP request it receives is destined for it. For every InARP request, the receiving station should format a proper response using the source addresses from the request as the target addresses of the response. If the station is unable or unwilling to reply, it ignores the request.

When the requesting station receives the InARP response, it may complete the ARP table entry and use the provided address information. Note: as with ARP, information learned via InARP may be aged or invalidated under certain circumstances.

## 7.1. Operation with Multi-Addressed Hosts

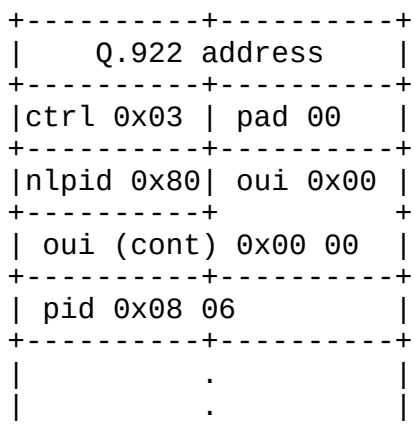
In the context of this discussion, a multi-addressed host will refer to a host that has multiple protocol addresses assigned to a single interface. If such a station receives an InARP request, it must choose one address with which to respond. To make such a selection, the receiving station must first look at the protocol address of the requesting station, and then respond with the protocol address corresponding to the network of the requester. For example, if the requesting station is probing for an IP address, the responding multi-addressed station should respond with an IP address which corresponds to the same subnet as the requesting station. If the station does not have an address that is appropriate for the request it should not respond. In the IP example, if the receiving station does not have an IP address assigned to the interface that is a part of the requested subnet, the receiving station would not respond.

A multi-addressed host should send an InARP request for each of the addresses defined for the given interface. It should be noted, however, that the receiving side may answer some or none of the requests depending on its configuration.

## 7.2. Protocol Operation Within Frame Relay

One case where Inverse ARP can be used is on a frame relay interface which supports signaling of DLCIs via a data link management interface. An InARP equipped station connected to such an interface will format an InARP request and address it to the new virtual circuit. If the other side supports InARP, it may return a response indicating the protocol address requested.

In a frame relay environment, InARP packets are encapsulated using the NLPID/SNAP format defined in [3] which indicates the ARP protocol. Specifically, the packet encapsulation will be as follows:




The format for an InARP request itself is defined by the following:

ar$hrd - 0x000F the value assigned to Frame Relay
ar$pro - protocol type for which you are searching
(i.e. IP = 0x0800)
ar$hln - 2,3, or 4 byte addressing length
ar$pln - byte length of protocol address for which you
are searching (for IP = 4)
ar$op - 8; InARP request
ar$sha - Q.922 [6] address of requesting station
ar$spa - protocol address of requesting station
ar$tha - Q.922 address of newly announced virtual circuit
ar$tpa - 0; This is what is being requested

The InARP response will be completed similarly.

ar$hrd - 0x000F the value assigned to Frame Relay
ar$pro - protocol type for which you are searching
(i.e. IP = 0x0800)
ar$hln - 2,3, or 4 byte addressing length
ar$pln - byte length of protocol address for which you
are searching (for IP = 4)
ar$op - 9; InARP response
ar$sha - Q.922 address of responding station
ar$spa - protocol address requested
ar$tha - Q.922 address of requesting station
ar$tpa - protocol address of requesting station

Note that the Q.922 addresses specified have the C/R, FECN, BECN, and DE bits set to zero.

Procedures for using InARP over a Frame Relay network are as follows:

Because DLCIs within most Frame Relay networks have only local significance, an end station will not have a specific DLCI assigned to itself. Therefore, such a station does not have an address to put into the InARP request or response. Fortunately, the Frame Relay network does provide a method for obtaining the correct DLCIs. The solution proposed for the locally addressed Frame Relay network below will work equally well for a network where DLCIs have global significance.

The DLCI carried within the Frame Relay header is modified as it traverses the network. When the packet arrives at its destination, the DLCI has been set to the value that, from the standpoint of the receiving station, corresponds to the sending station. For example, in figure 1 below, if station A were to send a message to station B, it would place DLCI 50 in the Frame Relay header. When station B received this message, however, the DLCI would have been modified by the network and would appear to B as DLCI 70.

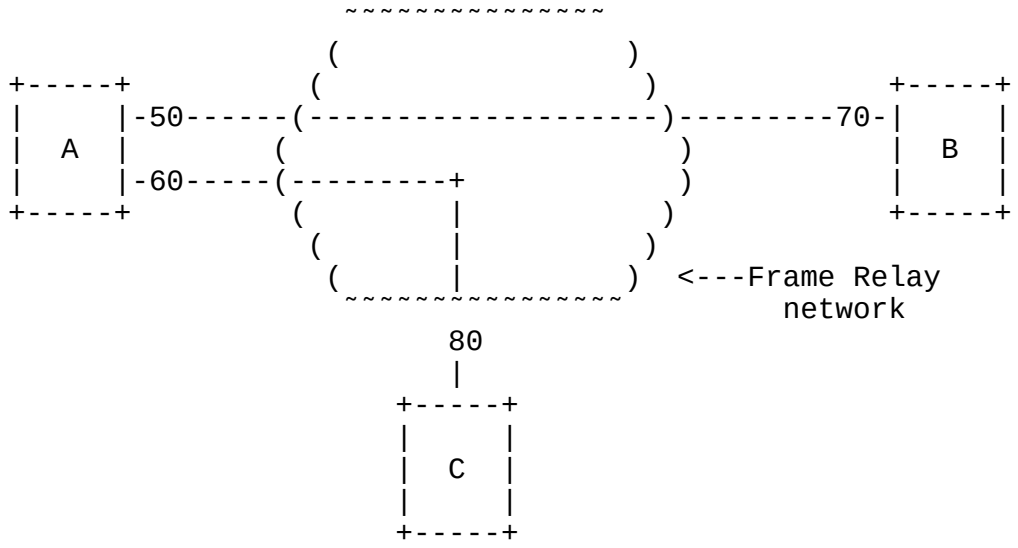


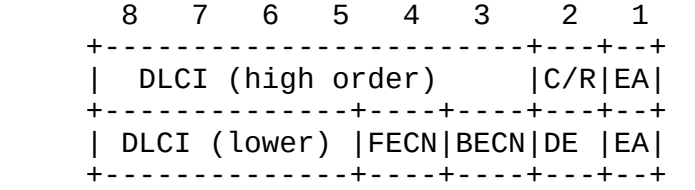

Figure 1

Lines between stations represent data link connections (DLCs). The numbers indicate the local DLCI associated with each connection.

DLCI to Q.922 Address Table for Figure 1

| DLCI (decimal) | Q.922 address (hex) |
|---|---|
| 50 | 0x0C21 |
| 60 | 0x0CC1 |
| 70 | 0x1061 |
| 80 | 0x1401 |

For authoritative description of the correlation between DLCI and Q.922 [6] addresses, the reader should consult that specification. A summary of the correlation is included here for convenience. The translation between DLCI and Q.922 address is based on a two byte address length using the Q.922 encoding format. The format is:




For InARP, the FECN, BECN, C/R and DE bits are assumed to be 0.

When an InARP message reaches a destination, all hardware addresses will be invalid. The address found in the frame header will, however, be correct. Though it does violate the purity of layering, Frame Relay may use the address in the header as the sender hardware address. It should also be noted that the target hardware address, in both the InARP request and response, will also be invalid. This should not cause problems since InARP does not rely on these fields and in fact, an implementation may zero fill or ignore the target hardware address field entirely.

Using figure 1 as an example, station A may use Inverse ARP to discover the protocol address of the station associated with its DLCI 50. The Inverse ARP request would be as follows:

```
InARP Request from A (DLCI 50)
ar$op   8       (InARP request)
ar$sha  unknown
ar$spa  pA
ar$tha  0x0C21  (DLCI 50)
ar$tpa  unknown
```

When Station B receives this packet, it will modify the source hardware address with the Q.922 address from the Frame Relay header. This way, the InARP request from A will become:

```
ar$op   8       (InARP request)
ar$sha  0x1061  (DLCI 70)
ar$spa  pA
ar$tha  0x0C21  (DLCI 50)
ar$tpa  unknown.
```

Station B will format an Inverse ARP response and send it to station A:

```
ar$op   9       (InARP response)
ar$sha  unknown
ar$spa  pB
ar$tha  0x1061  (DLCI 70)
ar$tpa  pA
```

The source hardware address is unknown and when the response is received, station A will extract the address from the Frame Relay header and place it in the source hardware address field. Therefore, the response will become:

```
ar$op   9       (InARP response)
ar$sha  0x0C21  (DLCI 50)
ar$spa  pB
ar$tha  0x1061  (DLCI 70)
ar$tpa  pA
```

This means that the Frame Relay interface must only intervene in the processing of incoming packets.

Also, see [3] for a description of similar procedures for using ARP [1] and RARP [4] with Frame Relay.

## 8. Security Considerations

This document specifies a functional enhancement to the ARP family of protocols, and is subject to the same security constraints that affect ARP and similar address resolution protocols. Because authentication is not a part of ARP, there are known security issues relating to its use (e.g., host impersonation). No additional security mechanisms have been added to the ARP family of protocols by this document.

## 9. References

[1] Plummer, D., "An Ethernet Address Resolution Protocol - or - Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, November 1982.

[2] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994. See also: http://www.iana.org/numbers.html

[3] Bradley, T., Brown, C., and A. Malis, "Multiprotocol Interconnect over Frame Relay", RFC 1490, July 1993.

[4] Finlayson, R., Mann, R., Mogul, J., and M. Theimer, "A Reverse Address Resolution Protocol", STD 38, RFC 903, June 1984.

[5] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[6] Information technology - Telecommunications and Information Exchange between systems - Protocol Identification in the Network Layer, ISO/IEC TR 9577: 1992.

## 10. Authors' Addresses

Terry Bradley
Avici Systems, Inc.
12 Elizabeth Drive
Chelmsford, MA 01824

Phone: (978) 250-3344
EMail: tbradley@avici.com

Caralyn Brown
Consultant

EMail: cbrown@juno.com

Andrew Malis
Ascend Communications, Inc.
1 Robbins Road
Westford, MA 01886

Phone: (978) 952-7414
EMail: malis@ascend.com

## 11. Full Copyright Statement