Network Working Group
Request for Comments: 1294

T. Bradley
C. Brown
Wellfleet Communications, Inc.
A. Malis
BBN Communications
January 1992

Multiprotocol Interconnect over Frame Relay

1. Status of this Memo

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

2. Abstract

This memo describes an encapsulation method for carrying network interconnect traffic over a Frame Relay backbone. It covers aspects of both Bridging and Routing. Systems with the ability to transfer both this encapsulation method, and others must have a priori knowledge of which virtual circuits will carry which encapsulation method and this encapsulation must only be used over virtual circuits that have been explicitly configured for its use.

3. Acknowledgements

Comments and contributions from many sources, especially those from Ray Samora of Proteon, Ken Rehbehn of Netrix Corporation, Fred Baker and Charles Carvalho of Advanced Computer Communications and Mostafa Sherif of AT&T have been incorporated into this document. Special thanks to Dory Leifer of University of Michigan for his contributions to the resolution of fragmentation issues. This document could not have been completed without the expertise of the IP over Large Public Data Networks working group of the IETF.

4. Conventions

The following language conventions are used in the items of specification in this document:

- Must, Shall or Mandatory -- the item is an absolute requirement of the specification.
- Should or Recommended -- the item should generally be followed for all but exceptional circumstances.

- o May or Optional -- the item is truly optional and may be followed or ignored according to the needs of the implementor.

5. Introduction

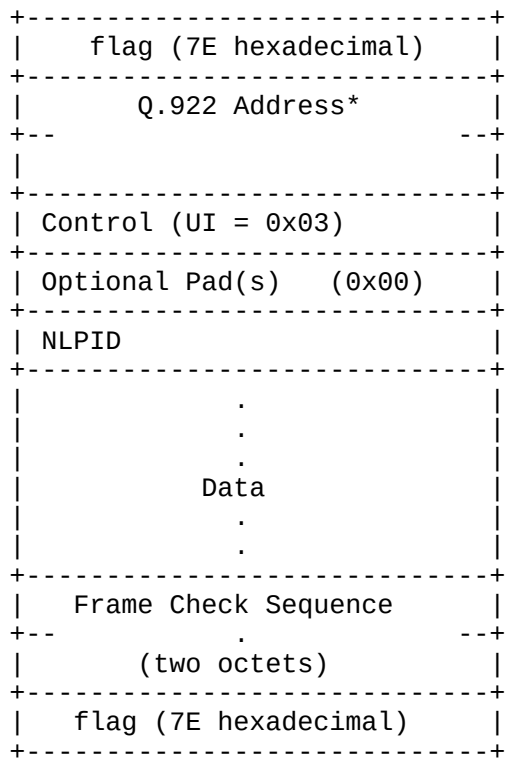
The following discussion applies to those devices which serve as end stations (DTEs) on a public or private Frame Relay network (for example, provided by a common carrier or PTT). It will not discuss the behavior of those stations that are considered a part of the Frame Relay network (DCEs) other than to explain situations in which the DTE must react.

The Frame Relay network provides a number of virtual circuits that form the basis for connections between stations attached to the same Frame Relay network. The resulting set of interconnected devices forms a private Frame Relay group which may be either fully interconnected with a complete "mesh" of virtual circuits, or only partially interconnected. In either case, each virtual circuit is uniquely identified at each Frame Relay interface by a Data Link Connection Identifier (DLCI). In most circumstances DLCIs have strictly local significance at each Frame Relay interface.

The specifications in this document are intended to apply to both switched and permanent virtual circuits.

6. Frame Format

All protocols must encapsulate their packets within a Q.922 Annex A frame [1,2]. Additionally, frames shall contain information necessary to identify the protocol carried within the Protocol Data Unit (PDU), thus allowing the receiver to properly process the incoming packet. The format shall be as follows:



* Q.922 addresses, as presently defined, are two octets and contain a 10-bit DLCI. In some networks Q.922 addresses may optionally be increased to three or four octets.

The control field is the Q.922 control field. The UI (0x03) value is used unless it is negotiated otherwise. The use of XID (0xAF or 0xBF) is permitted and is discussed later.

The pad field is an optional field used to align the remainder of the frame to a convenient boundary for the sender. There may be zero or more pad octets within the pad field and all must have a value of zero.

The Network Level Protocol ID (NLPID) field is administered by ISO and CCITT. It contains values for many different protocols including IP, CLNP and IEEE Subnetwork Access Protocol (SNAP)[10]. This field tells the receiver what encapsulation or what protocol follows. Values for this field are defined in ISO/IEC TR 9577 [3]. A NLPID value of 0x00 is defined within ISO/IEC TR 9577 as the Null Network Layer or Inactive Set. Since it cannot be distinguished from a pad field, and because it has no significance within the context of this

encapsulation scheme, a NLPID value of 0x00 is invalid under the Frame Relay encapsulation. The known NLPID values are listed in the Appendix.

For full interoperability with older Frame Relay encapsulation formats, a station may implement section 15, Backward Compatibility.

There is no commonly implemented maximum frame size for Frame Relay. A network must, however, support at least a 262 octet maximum. Generally, the maximum will be greater than or equal to 1600 octets, but each Frame Relay provider will specify an appropriate value for its network. A Frame Relay DTE, therefore, must allow the maximum acceptable frame size to be configurable.

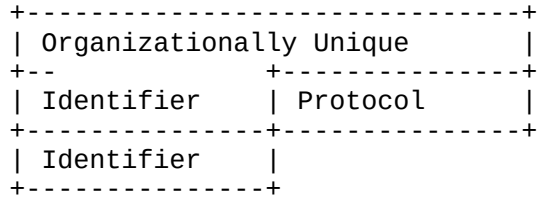
The minimum frame size allowed for Frame Relay is five octets between the opening and closing flags.

7. Interconnect Issues

There are two basic types of data packets that travel within the Frame Relay network, routed packets and bridged packets. These packets have distinct formats and therefore, must contain an indication that the destination may use to correctly interpret the contents of the frame. This indication is embedded within the NLPID and SNAP header information.

For those protocols that do not have a NLPID already assigned, it is necessary to provide a mechanism to allow easy protocol identification. There is a NLPID value defined indicating the presence of a SNAP header.

A SNAP header is of the form



All stations must be able to accept and properly interpret both the NLPID encapsulation and the SNAP header encapsulation for a routed packet.

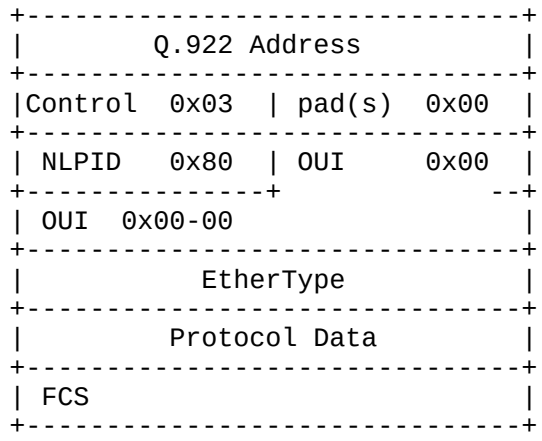
The three-octet Organizationally Unique Identifier (OUI) identifies an organization which administers the meaning of the Protocol Identifier (PID) which follows. Together they identify a distinct

protocol. Note that OUI 0x00-00-00 specifies that the following PID is an EtherType.

7.1. Routed Frames

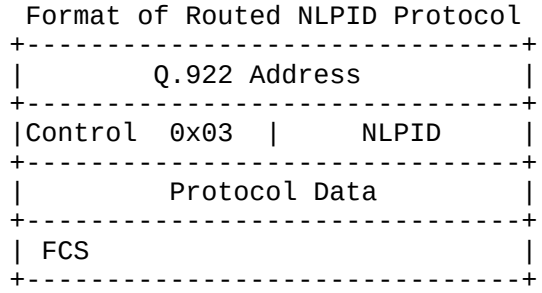
Some protocols will have an assigned NLPID, but because the NLPID numbering space is so limited many protocols do not have a specific NLPID assigned to them. When packets of such protocols are routed over Frame Relay networks they are sent using the NLPID 0x80 (which indicates a SNAP follows), OUI 0x00-00-00 (which indicates an EtherType follows), and the EtherType of the protocol in use.

Format of Routed Frames

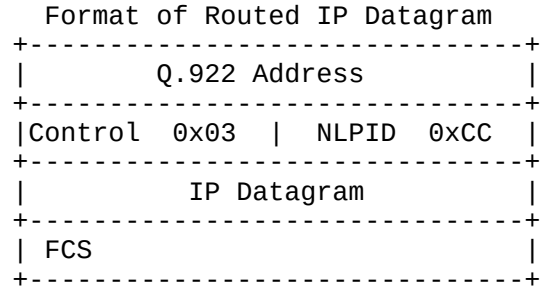


In the few cases when a protocol has an assigned NLPID (see appendix), 48 bits can be saved using the format below:

Format of Routed NLPID Protocol



In the particular case of an Internet IP datagram, the NLPID is 0xCC.



7.2. Bridged Frames

The second type of Frame Relay traffic is bridged packets. These packets are encapsulated using the NLPID value of 0x80 indicating SNAP and the following SNAP header identifies the format of the bridged packet. The OUI value used for this encapsulation is the 802.1 organization code 0x00-80-C2. The following two octets (PID) specify the form of the MAC header, which immediately follows the SNAP header. Additionally, the PID indicates whether the original FCS is preserved within the bridged frame.

The 802.1 organization has reserved the following values to be used with Frame Relay:

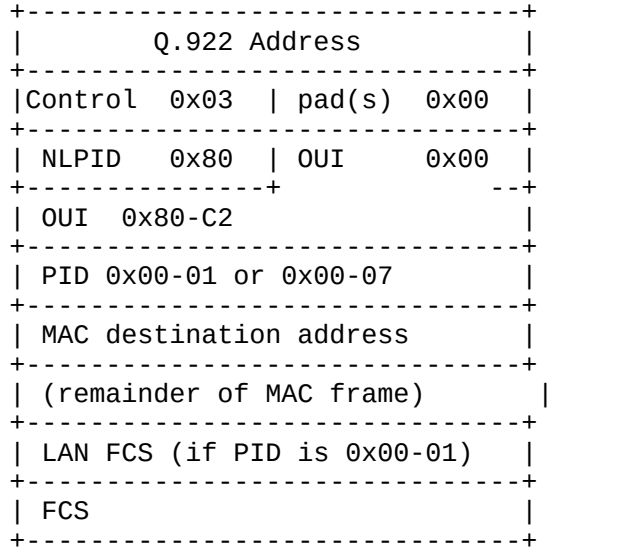
PID Values for OUI 0x00-80-C2

with preserved FCS	w/o preserved FCS	Media
0x00-01	0x00-07	802.3/Ethernet
0x00-02	0x00-08	802.4
0x00-03	0x00-09	802.5
0x00-04	0x00-0A	FDDI
0x00-05	0x00-0B	802.6

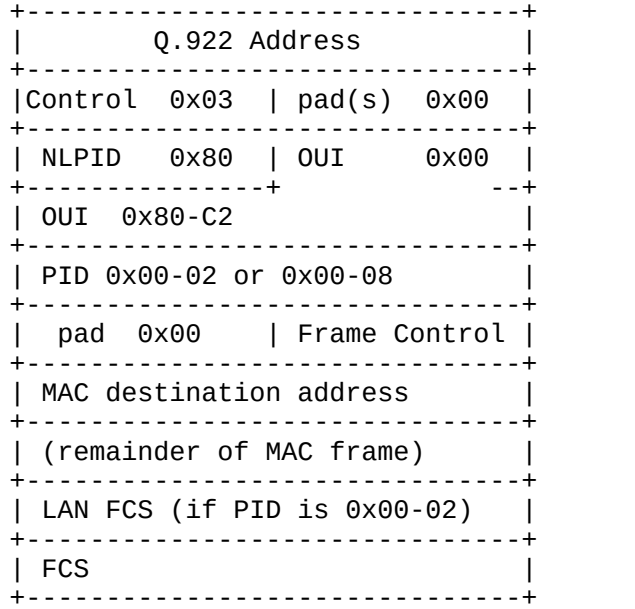
In addition, the PID value 0x00-0E, when used with OUI 0x00-80-C2, identifies Bridged Protocol Data Units (BPDUs).

A packet bridged over Frame Relay will, therefore, have one of the following formats:

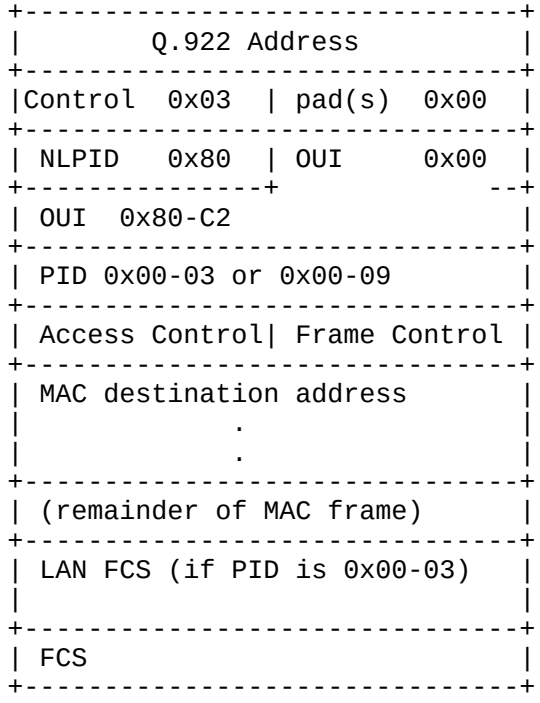
Format of Bridged Ethernet/802.3 Frame



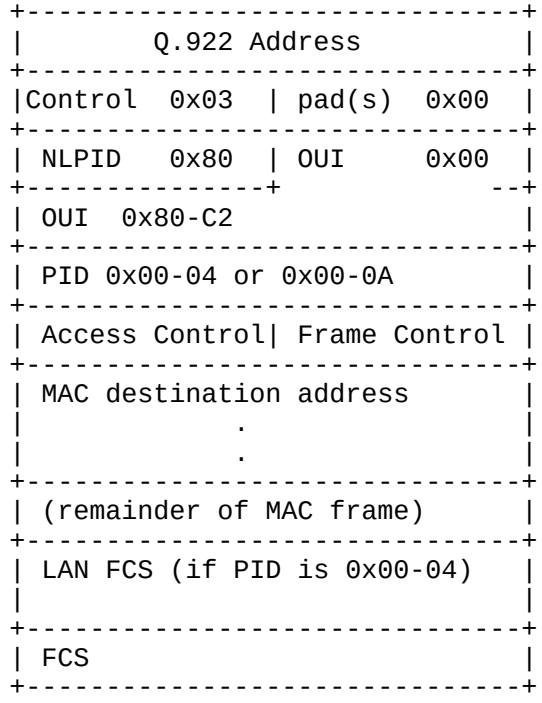
Format of Bridged 802.4 Frame

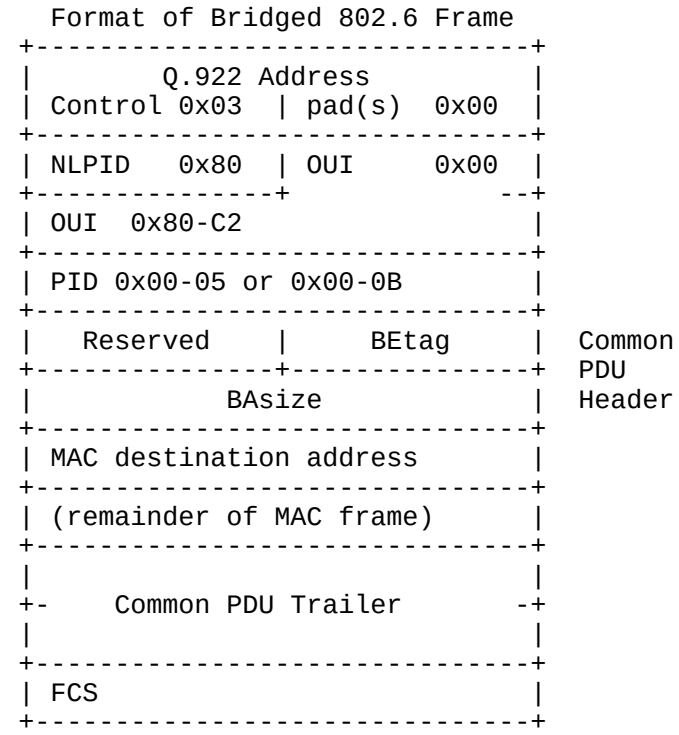


Format of Bridged 802.5 Frame



Format of Bridged FDDI Frame

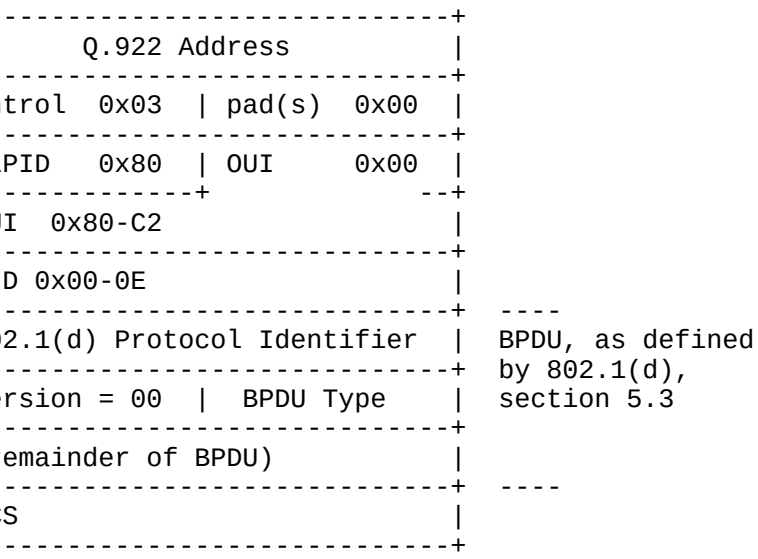




The Common Protocol Data Unit (PDU) Header and Trailer are conveyed to allow pipelining at the egress bridge to an 802.6 subnetwork. Specifically, the Common PDU Header contains the BAsize field, which contains the length of the PDU. If this field is not available to the egress 802.6 bridge, then that bridge cannot begin to transmit the segmented PDU until it has received the entire PDU, calculated the length, and inserted the length into the BAsize field. If the field is available, the egress 802.6 bridge can extract the length from the BAsize field of the Common PDU Header, insert it into the corresponding field of the first segment, and immediately transmit the segment onto the 802.6 subnetwork. Thus, the bridge can begin transmitting the 802.6 PDU before it has received the complete PDU.

One should note that the Common PDU Header and Trailer of the encapsulated frame should not be simply copied to the outgoing 802.6 subnetwork because the encapsulated BEtag value may conflict with the previous BEtag value transmitted by that bridge.

Format of BPDU Frame



8. Data Link Layer Parameter Negotiation

Frame Relay stations may choose to support the Exchange Identification (XID) specified in Appendix III of Q.922 [1]. This XID exchange allows the following parameters to be negotiated at the initialization of a Frame Relay circuit: maximum frame size N201, retransmission timer T200, and the maximum number of outstanding I frames K.

A station may indicate its unwillingness to support acknowledged mode multiple frame operation by specifying a value of zero for the maximum window size, K.

If this exchange is not used, these values must be statically configured by mutual agreement of Data Link Connection (DLC) endpoints, or must be defaulted to the values specified in Section 5.9 of Q.922:

N201: 260 octets

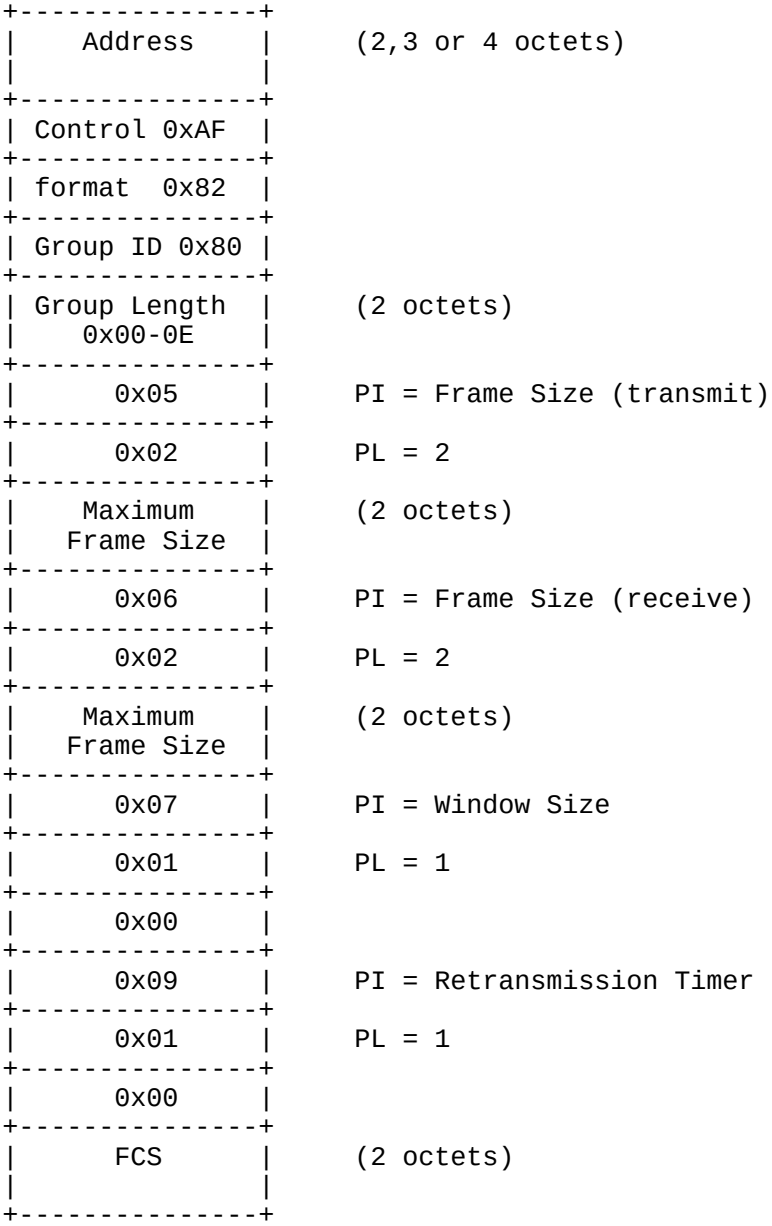
K: 3 for a 16 Kbps link,
7 for a 64 Kbps link,
32 for a 384 Kbps link,
40 for a 1.536 Mbps or above link

T200: 1.5 seconds [see Q.922 for further details]

If a station supporting XID receives an XID frame, it shall respond with an XID response. In processing an XID, if the remote maximum frame size is smaller than the local maximum, the local system shall reduce the maximum size it uses over this DLC to the remotely specified value. Note that this shall be done before generating a response XID.

The following diagram describes the use of XID to specify non-use of acknowledged mode multiple frame operation.

Non-use of Acknowledged Mode Multiple Frame Operation



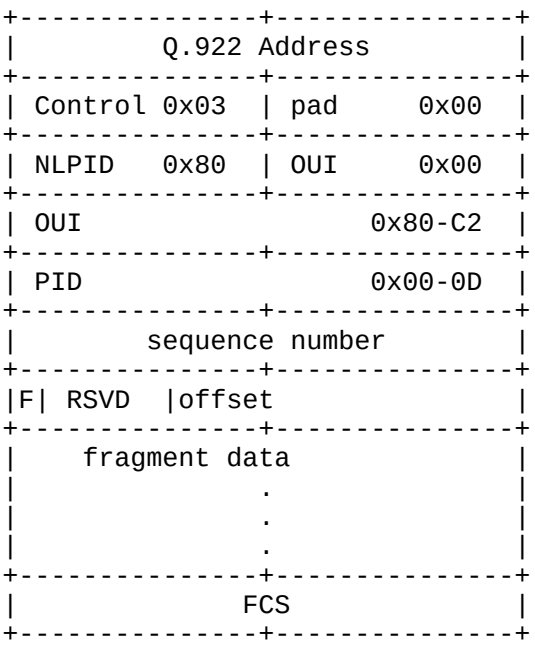
9. Fragmentation Issues

Fragmentation allows the exchange of packets that are greater than the maximum frame size supported by the underlying network. In the case of Frame Relay, the network may support a maximum frame size as small as 262 octets. Because of this small maximum size, it is advantageous to support fragmentation and reassembly.

Unlike IP fragmentation procedures, the scope of Frame Relay fragmentation procedure is limited to the boundary (or DTEs) of the Frame Relay network.

The general format of fragmented packets is the same as any other encapsulated protocol. The most significant difference being that the fragmented packet will contain the encapsulation header. That is, a packet is first encapsulated (with the exception of the address and control fields) as defined above. Large packets are then broken up into frames appropriate for the given Frame Relay network and are encapsulated using the Frame Relay fragmentation format. In this way, a station receiving fragments may reassemble them and then put the reassembled packet through the same processing path as a packet that had not been fragmented.

Within Frame Relay fragments are encapsulated using the SNAP format with an OUI of 0x00-80-C2 and a PID of 0x00-0D. Individual fragments will, therefore, have the following format:



The sequence field is a two octet identifier that is incremented every time a new complete message is fragmented. It allows detection of lost frames and is set to a random value at initialization.

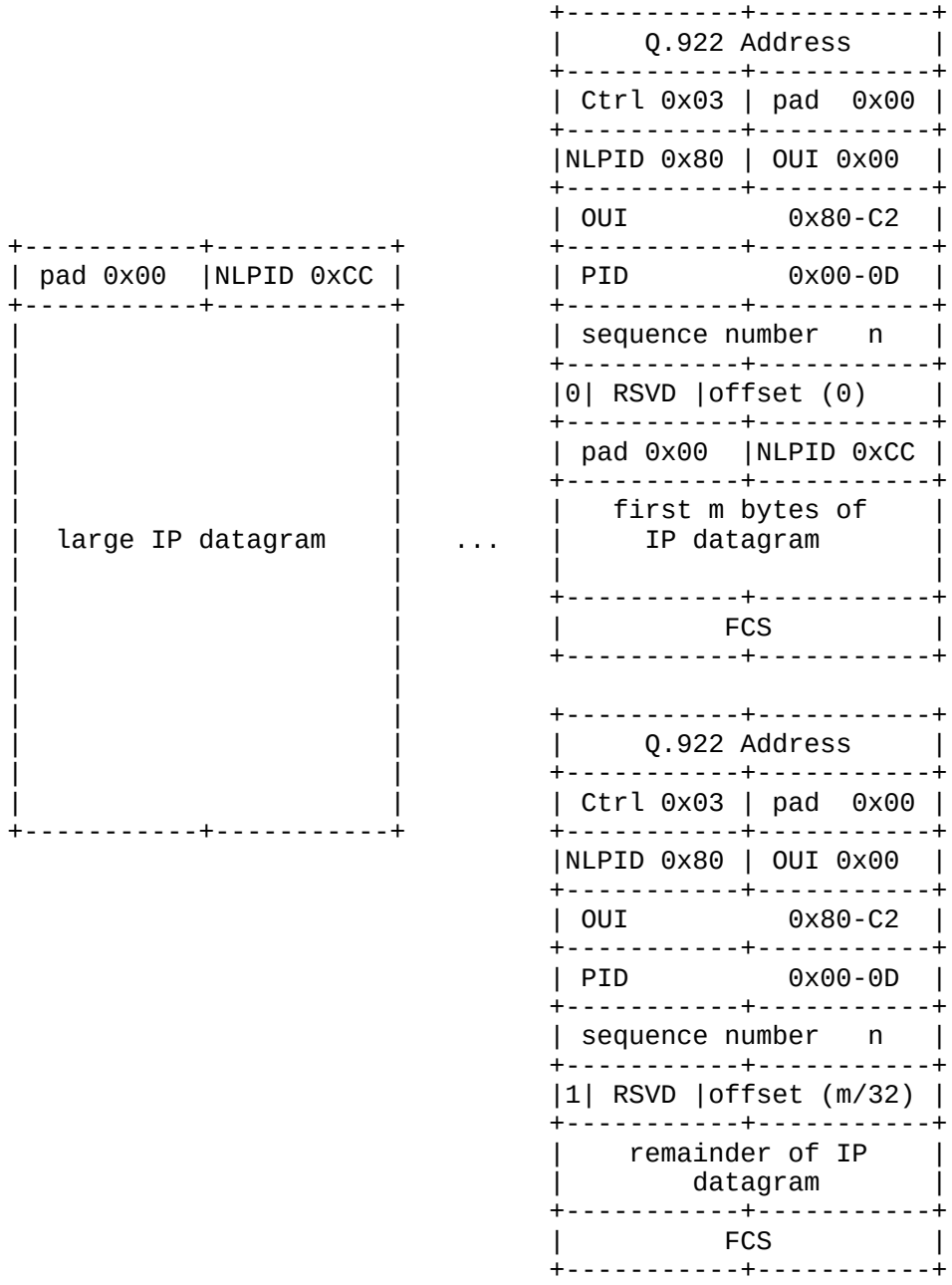
The reserved field is 4 bits long and is not currently defined. It must be set to 0.

The final bit is a one bit field set to 1 on the last fragment and set to 0 for all other fragments.

The offset field is an 11 bit value representing the logical offset of this fragment in bytes divided by 32. The first fragment must have an offset of zero.

The following figure shows how a large IP datagram is fragmented over Frame Relay. In this example, the complete datagram is fragmented into two Frame Relay frames.

Frame Relay Fragmentation Example



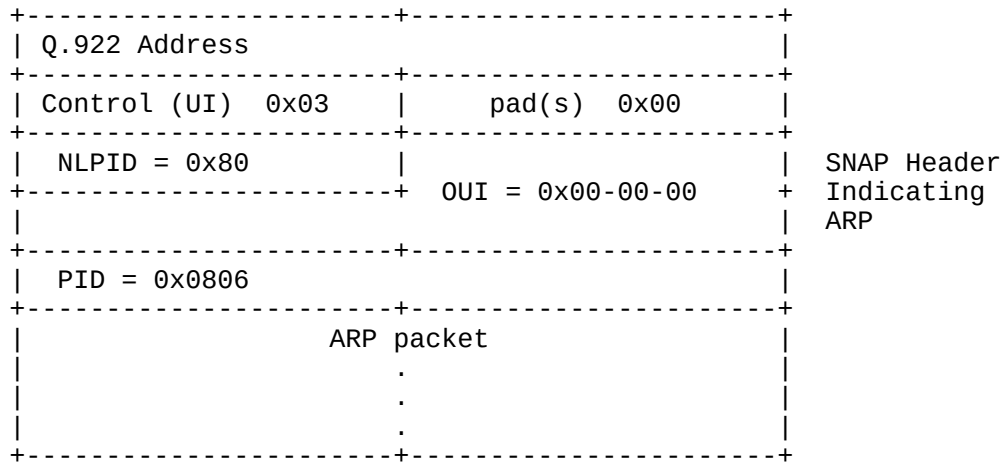
Fragments must be sent in order starting with a zero offset and ending with the final fragment. These fragments must not be

interrupted with other packets or information intended for the same DLC. An end station must be able to re-assemble up to 2K octets and is suggested to support up to 8K octet re-assembly. If at any time during this re-assembly process, a fragment is corrupted or a fragment is missing, the entire message is dropped. The upper layer protocol is responsible for any retransmission in this case.

This fragmentation algorithm is not intended to reliably handle all possible failure conditions. As with IP fragmentation, there is a small possibility of reassembly error and delivery of an erroneous packet. Inclusion of a higher layer checksum greatly reduces this risk.

10. Address Resolution

There are situations in which a Frame Relay station may wish to dynamically resolve a protocol address. Address resolution may be accomplished using the standard Address Resolution Protocol (ARP) [6] encapsulated within a SNAP encoded Frame Relay packet as follows:



Where the ARP packet has the following format and values:

```
Data:
  ar$hrd   16 bits     Hardware type
  ar$pro   16 bits     Protocol type
  ar$hln    8 bits     Octet length of hardware address (n)
  ar$pln    8 bits     Octet length of protocol address (m)
  ar$op    16 bits     Operation code (request or reply)
  ar$sha   noctets     source hardware address
  ar$spa   moctets     source protocol address
  ar$tha   noctets     target hardware address
  ar$tpa   moctets     target protocol address
```

ar$hrd - assigned to Frame Relay is 15 decimal (0x000F) [7].

ar$pro - see assigned numbers for protocol ID number for the protocol using ARP. (IP is 0x0800).

ar$hln - length in bytes of the address field (2, 3, or 4)

ar$pln - protocol address length is dependent on the protocol (ar$pro) (for IP ar$pln is 4).

ar$op - 1 for request and 2 for reply.

ar$sha - Q.922 source hardware address, with C/R, FECN, BECN, and DE set to zero.

ar$tha - Q.922 target hardware address, with C/R, FECN, BECN, and DE set to zero.

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 ARP request or reply. 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.

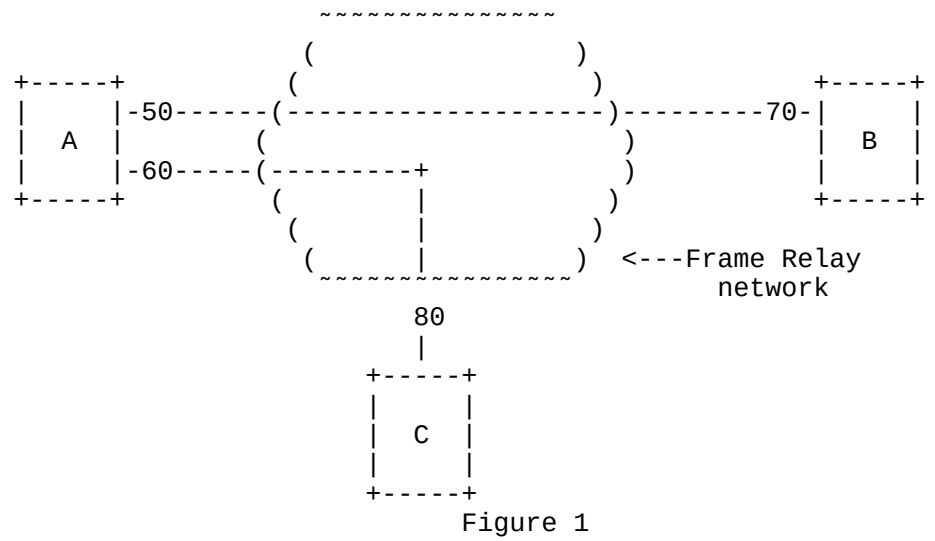


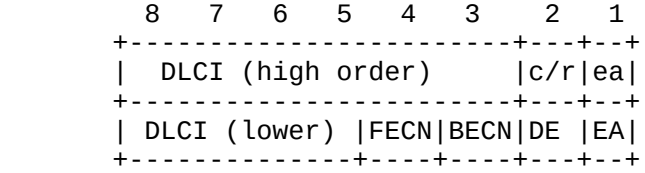
Figure 1

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

If you know about frame relay, you should understand the corrolation between DLCI and Q.922 address. For the uninitiated, 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 ARP and its variants, the FECN, BECN, C/R and DE bits are assumed to be 0.

When an ARP 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 ARP request and reply, will also be invalid. This should not cause problems since ARP does not rely on these fields and in fact, an implementation may zero fill or ignore the target hardware address field entirely.

As an example of how this address replacement scheme may work, refer to figure 1. If station A (protocol address pA) wished to resolve the address of station B (protocol address pB), it would format an ARP request with the following values:

```
ARP request from A
  ar$op     1 (request)
  ar$sha    unknown
  ar$spa    pA
  ar$tha    undefined
  ar$tpa    pB
```

Because station A will not have a source address associated with it, the source hardware address field is not valid. Therefore, when the ARP packet is received, it must extract the correct address from the Frame Relay header and place it in the source hardware address field. This way, the ARP request from A will become:

```
ARP request from A as modified by B
  ar$op     1 (request)
  ar$sha    0x1061 (DLCI 70) from Frame Relay header
  ar$spa    pA
  ar$tha    undefined
  ar$tpa    pB
```

Station B's ARP will then be able to store station A's protocol address and Q.922 address association correctly. Next, station B will form a reply message. Many implementations simply place the source addresses from the ARP request into the target addresses and then fills in the source addresses with its addresses. In this case, the ARP response would be:

```
ARP response from B
  ar$op     2 (response)
  ar$sha    unknown
  ar$spa    pB
  ar$tha    0x1061 (DLCI 70)
  ar$tpa    pA
```

Again, the source hardware address is unknown and when the request 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:

```
ARP response from B as modified by A
  ar$op     2 (response)
  ar$sha    0x0C21 (DLCI 50)
  ar$spa    pB
  ar$tha    0x1061 (DLCI 70)
  ar$tpa    pA
```

Station A will now correctly recognize station B having protocol address pB associated with Q.922 address 0x0C21 (DLCI 50).

Reverse ARP (RARP) [8] will work in exactly the same way. Still using figure 1, if we assume station C is an address server, the following RARP exchanges will occur:

```
RARP request from A              RARP request as modified by C
   ar$op  3 (RARP request)          ar$op  3  (RARP request)
   ar$sha unknown                   ar$sha 0x1401 (DLCI 80)
   ar$spa undefined                 ar$spa undefined
   ar$tha 0x0CC1 (DLCI 60)          ar$tha 0x0CC1 (DLCI 60)
   ar$tpa pC                        ar$tpa pC
```

Station C will then look up the protocol address corresponding to Q.922 address 0x1401 (DLCI 80) and send the RARP response.

```
RARP response from C             RARP response as modified by A
   ar$op  4  (RARP response)        ar$op  4 (RARP response)
   ar$sha unknown                   ar$sha 0x0CC1 (DLCI 60)
   ar$spa pC                        ar$spa pC
   ar$tha 0x1401 (DLCI 80)          ar$tha 0x1401 (DLCI 80)
   ar$tpa pA                        ar$tpa pA
```

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

In the absence of suitable multicast, ARP may still be implemented. To do this, the end station simply sends a copy of the ARP request through each relevant DLC, thereby simulating a broadcast.

The use of multicast addresses in a Frame Relay environment is presently under study by Frame Relay providers. At such time that the issues surrounding multicasting are resolved, multicast addressing may become useful in sending ARP requests and other "broadcast" messages.

Because of the inefficiencies of broadcasting in a Frame Relay environment, a new address resolution variation was developed. It is called Inverse ARP [11] and describes a method for resolving a protocol address when the hardware address is already known. In Frame Relay's case, the known hardware address is the DLCI. Using Inverse ARP for Frame Relay follows the same pattern as ARP and RARP use. That is the source hardware address is inserted at the receiving station.

In our 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:

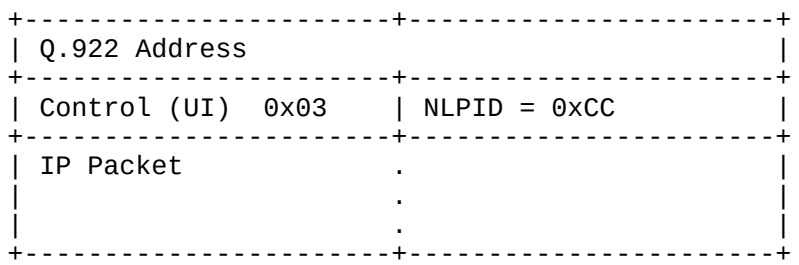
```
ar$op   8       (InARP request)
ar$sha  0x1061
ar$spa  pA
ar$tha  0x0C21
ar$tpa  unknown.
```

Station B will format an Inverse ARP response and send it to station A as it would for any ARP message.

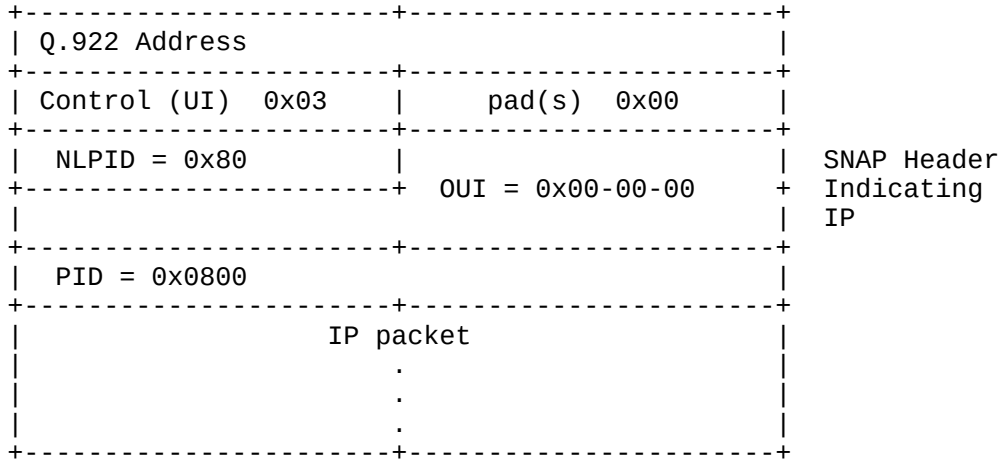
11. IP over Frame Relay

Internet Protocol [9] (IP) datagrams sent over a Frame Relay network conform to the encapsulation described previously. Within this context, IP could be encapsulated in two different ways.

1. NLPID value indicating IP



2. NLPID value indicating SNAP



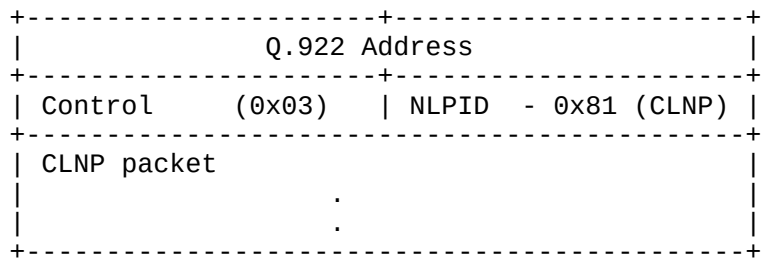
Although both of these encapsulations are supported under the given definitions, it is advantageous to select only one method as the appropriate mechanism for encapsulating IP data. Therefore, IP data shall be encapsulated using the NLPID value of 0xCC indicating IP as shown in option 1 above. This (option 1) is more efficient in transmission (48 fewer bits), and is consistent with the encapsulation of IP in X.25.

12. Other Protocols over Frame Relay

As with IP encapsulation, there are alternate ways to transmit various protocols within the scope of this definition. To eliminate the conflicts, the SNAP encapsulation is only used if no NLPID value is defined for the given protocol.

As an example of how this works, ISO CLNP has a NLPID defined (0x81). Therefore, the NLPID field will indicate ISO CLNP and the data packet

will follow immediately. The frame would be as follows:



13. Bridging in a Frame Relay network

A Frame Relay interface acting as a bridge must be able to flood, forward, and filter packets.

Flooding is performed by sending the packet to all possible destinations. In the Frame Relay environment this means sending the packet through each relevant DLC.

To forward a packet, a bridge must be able to associate a destination MAC address with a DLC. It is unreasonable and perhaps impossible to require bridges to statically configure an association of every possible destination MAC address with a DLC. Therefore, Frame Relay bridges must provide enough information to allow a Frame Relay interface to dynamically learn about foreign destinations beyond the set of Frame Relay stations.

To accomplish dynamic learning, a bridged packet shall conform to the encapsulation described within section 7. In this way, the receiving Frame Relay interface will know to look into the bridged packet and learn the association between foreign destination and Frame Relay station.

14. For Future Study

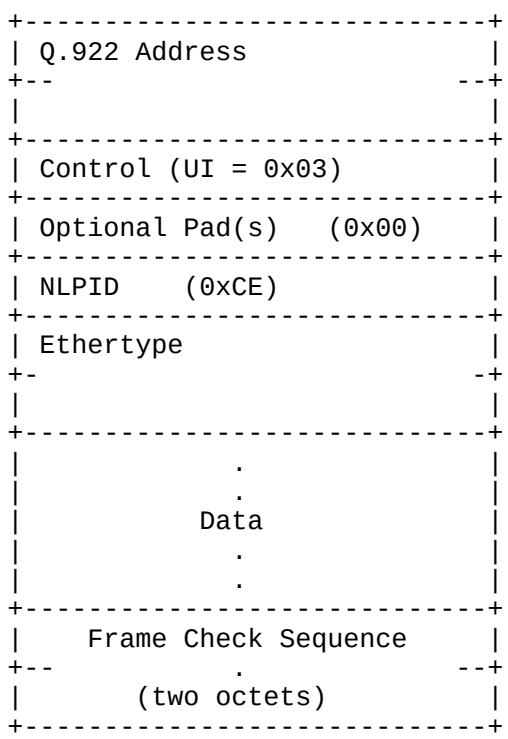
It may be desirable for the two ends of a connection to have the capability to negotiate end-to-end configuration and service parameters. The actual protocol and parameters to be negotiated will be a topic of future RFCs.

15. Backward Compatibility

This section is included in this RFC for completeness only. It is not intended to suggest additional requirements.

Some existing Frame Relay stations use the NLPID value of 0xCE to

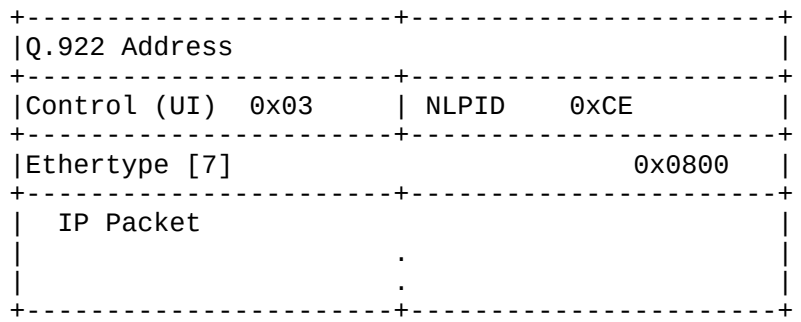
indicate an escape to Ethernet Packet Types as defined in the latest version of the Assigned Numbers (RFC-1060) [7]. In this case, the frame will have the following format:



The Ethertype field is a 16-bit value used to identify a protocol type for the following PDU.

In order to be fully interoperable with stations that use this encoding, Frame Relay stations may recognize the NLPID value of 0xCE and interpret the following two byte Ethertype. It is never necessary to generate this encapsulation format only to properly interpret it’s meaning.

For example, IP encapsulated with this NLPID value will have the following format:



16. Appendix

List of Known NLPIDs

0x00	Null Network Layer or Inactive Set (not used with Frame Relay)
0x80	SNAP
0x81	ISO CLNP
0x82	ISO ESIS
0x83	ISO ISIS
0xCC	Internet IP
0xCE	EtherType - unofficial temporary use

List of PIDs of OUI 00-80-C2

with preserved FCS	w/o preserved FCS	Media
0x00-01	0x00-07	802.3/Ethernet
0x00-02	0x00-08	802.4
0x00-03	0x00-09	802.5
0x00-04	0x00-0A	FDDI
0x00-05	0x00-0B	802.6
0x00-0D		Fragments
0x00-0E		BPDUs

17. References

[1] International Telegraph and Telephone Consultative Committee, "ISDN Data Link Layer Specification for Frame Mode Bearer Services", CCITT Recommendation Q.922, 19 April 1991 .

[2] American National Standard For Telecommunications - Integrated Services Digital Network - Core Aspects of Frame Protocol for Use with Frame Relay Bearer Service, ANSI T1.618-1991, 18 June 1991.

[3] Information technology - Telecommunications and Information Exchange between systems - Protocol Identification in the Network Layer, ISO/IEC TR 9577: 1990 (E) 1990-10-15.

[4] Baker, Fred, "Point to Point Protocol Extensions for Bridging", Point to Point Working Group, RFC-1220, April 1991.

[5] International Standard, Information Processing Systems - Local Area Networks - Logical Link Control, ISO 8802-2: 1989 (E), IEEE Std 802.2-1989, 1989-12-31.

[6] Plummer, David C., An Ethernet Address Resolution Protocol", RFC-826, November 1982.

[7] Reynolds, J. and Postel, J., "Assigned Numbers", RFC-1060, ISI, March 1990.

[8] Finlayson, Mann, Mogul, Theimer, "A Reverse Address Resolution Protocol", RFC-903, Stanford University, June 1984.

[9] Postel, J. and Reynolds, J., "A Standard for the Transmission of IP Datagrams over IEEE 802 Networks", RFC-1042, ISI, February 1988.

[10] IEEE, "IEEE Standard for Local and Metropolitan Area Networks: Overview and architecture", IEEE Standards 802-1990.

[11] Bradley, T., and C. Brown, "Inverse Address Resolution Protocol", RFC-1293, Wellfleet Communications, Inc., January 1992.

18. Security Considerations

Security issues are not addressed in this memo.

19. Authors' Addresses

Terry Bradley
Wellfleet Communications, Inc.
15 Crosby Drive
Bedford, MA 01730

Phone: (617) 275-2400

Email: tbradley@wellfleet.com

Caralyn Brown
Wellfleet Communications, Inc.
15 Crosby Drive
Bedford, MA 01730

Phone: (617) 275-2400

Email: cbrown@wellfleet.com

Andrew G. Malis
BBN Communications
150 CambridgePark Drive
Cambridge, MA 02140

Phone: (617) 873-3419

Email: malis@bbn.com