                 Internet Protocol on Network Systems HYPERchannel
                             Protocol Specification

STATUS OF THIS MEMO

   The intent of this document is to provide a complete discussion of
   the protocols and techniques used to embed DoD standard Internet
   Protocol datagrams (and its associated higher level protocols) on
   Network Systems Corporation's HYPERchannel [1] equipment.
   Distribution of this memo is unlimited.

   This document is intended for network planners and implementors who
   are already familiar with the TCP/IP protocol suite and the
   techniques used to carry TCP/IP traffic on common networks such as
   the DDN or Ethernet. No great familiarity with NSC products is
   assumed; an appendix is devoted to a review of NSC technologies and
   protocols.

   At the time of this first RFC edition, the contents of this document
   has already been reviewed by about a dozen vendors and users active
   in the use of TCP/IP on HYPERchannel media. Comments and suggestions
   are still welcome (and implementable,) however.

   Any comments or questions on this specification may be directed to:

      Ken Hardwick
      Director, Software Technology
      Network Systems Corporation MS029
      7600 Boone Avenue North
      Brooklyn Park, MN 55428

      Phone: (612) 424-1607

      John Lekashman
      Nasa Ames Research Center. NAS/GE
      MS 258-6
      Moffett Field, CA, 94035
      lekash@orville.nas.nasa.gov

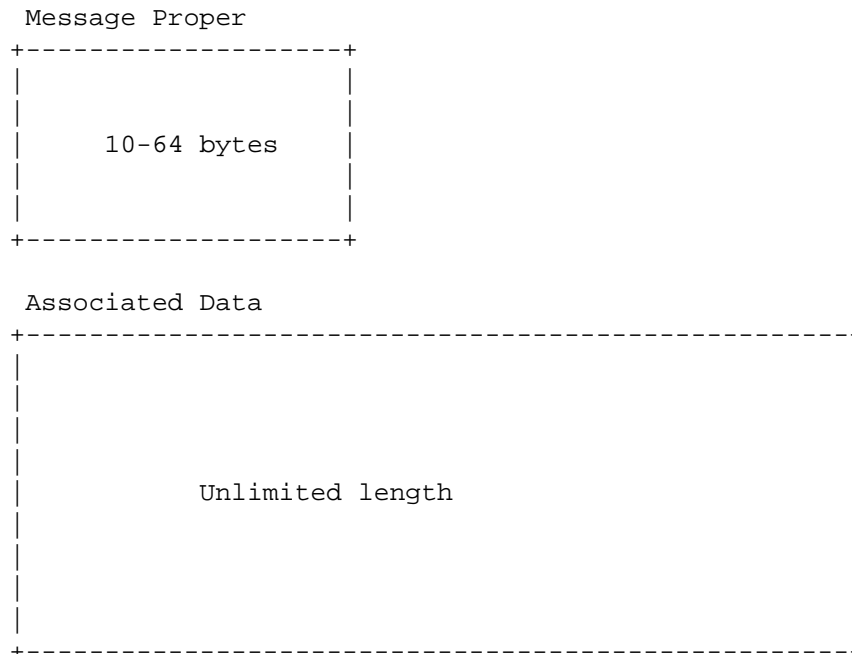   Phone: (415) 694-4359

TABLE OF CONTENTS

GOALS OF THIS DOCUMENT

    In this document, there are four major technical objectives:

    1. To bless a "de facto" standard for IP on HYPERchannel that has
        been implemented by Tektronix, Cray, NASA Ames, and others.
        We are attempting to resolve some interoperability problems with
        this standard so as to minimize the changes to existing IP on
        HYPERchannel software. If any ambiguities remain in the de facto
        standard, we wish to assist in their resolution.

    2. To address larger networks, NSC's newer network products are
        moving to a 32-bit address from the current 16-bit TO address.
        This document would introduce the addressing extension to the
        user community and specify how IP datagrams would work in the
        new addressing mode.

    3. To define an Address Resolution Protocol for HYPERchannel and
        other NSC products. It is probably well known that current NSC
        products do not support the broadcast modes that make ARP
        particularly useful. However, many have expressed interest in
        "ARP servers" at a known network address. These servers could
        fade away as NSC products with broadcast capability come into
        existence. Host drivers that can generate and recognize this
        ARP protocol would be prepared to take advantage of it as the
        pieces fall into place.

    4. Part of this effort is to standardize the unofficial "message
        type" field that reserves byte 8 of the HYPERchannel network
        message. To permit better interoperability, NSC will initiate a
        "network protocol registry" where any interested party may
        obtain a unique value in byte 8 (or bytes 8 and 9) for their own
        public, private, commercial or proprietary protocol. Lists of
        assigned protocol type numbers and their "owners" will be
        periodically published by NSC and would be available to
        interested parties.

BASIC HYPERCHANNEL NETWORK MESSAGES

   Unlike most datagram delivery systems, the HYPERchannel network
   message consists of two parts:

```
            Message Proper
            +-------------------+
            |                   |
            |                   |
            |     10-64 bytes   |
            |                   |
            |                   |
            +-------------------+


            Associated Data
            +---------------------------------------------------+
            |                                                   |
            |                                                   |
            |                                                   |
            |                                                   |
            |           Unlimited length                        |
            |                                                   |
            |                                                   |
            |                                                   |
            |                                                   |
            +---------------------------------------------------+
```

   The first part is a message header that can be up to 64 bytes in
   length. The first 10 bytes contain information required for the
   delivery of the entire message, and the remainder can be used by
   higher level protocols. The second part of the message, the
   "Associated Data," can be optionally included with the message
   proper. In most cases (transmission over HYPERchannel A trunks), the
   length of the associated data is literally unlimited. Others (such
   as HYPERchannel B or transmission within a local HYPERchannel A A400
   adapter) limit the size of the Associated Data to 4K bytes. If the
   information sent can be contained within the Message Proper, then the
   Associated Data need not be sent.

   HYPERchannel lower link protocols treat messages with and without
   Associated Data quite differently; "Message only" transmissions are
   sent using abbreviated protocols and can be queued in the receiving
   network adapter, thus minimizing the elapsed time needed to send and
   receive the messages. When associated data is provided, the
   HYPERchannel A adapters free their logical resources towards driving
   the host interface and coaxial trunks.

BASIC (16-BIT ADDRESS) MESSAGE PROPER HEADER

   The first 10 bytes of the network Message Proper are examined by the
   network adapters to control delivery of the network message. Its
   format is as follows:

```
    byte   Message Proper
         +-----------------------------+-----------------------------+
      0  |        Trunks to Try        |        Message Flags        |
         |    TO trunks  |  FROM trunks |              |EXC|BST|A/D|
         +--------------+--------------+-----------------+---+---+---+
      2  |                         Access code                       |
         |                                                           |
         +-----------------------------+-----------------------------+
      4  |        Physical addr of     |                   | TO Port |
         |     destination adapter (TO) |                  | number  |
         +-----------------------------+-----------------------------+
      6  |   Physical addr of source   |                   |FROM port|
         |        adapter (FROM)       |                   | number  |
         +-----------------------------+-----------------------------+
      8  |                        Message type                       |
         |                                                           |
         +-----------------------------+-----------------------------+
     10  |                                                           |
         |          Available for higher level protocols             |
         |                                                           |
         |                                                           |
         +-----------------------------+-----------------------------+
```

TRUNKS TO TRY

   Consists of two four bit masks indicating which of four possible
   HYPERchannel A coaxial data trunks are to be used to transmit the
   message and to return it. If a bit in the mask is ON, then the
   adapter firmware will logically AND it with the mask of installed
   trunk interfaces and use the result as a candidate list of
   interfaces. Whenever one of the internal "frames" are sent to
   communicate with the destination adapter, the transmission hardware
   electronically selects the first non-busy trunk out of the list of
   candidates. Thus, selection of a data trunk is best performed by the
   adapter itself rather than by the host. "Dedicating" trunks to
   specific applications only makes sense in very critical real time
   applications such as streaming data directly from high speed
   overrunnable peripherals.

   A second Trunk mask is provided for the receiving adapter when it
   sends frames back to the transmitter, as it is possible to build
   "asymmetric" configurations of data trunks where trunk 1 on one box

is connected to the trunk 3 interface of a second. Such
configurations are strongly discouraged, but the addressing structure
supports it if needed.

The "trunks to try" field is only used by HYPERchannel A. To assure
maximum interoperability, a value of 0xFF should be placed in this
field to assure delivery over any technology. Other values should
only be used if the particular site hardware is so configured to not
be physically connected via those trunks.

MESSAGE FLAGS

Contains options in message delivery. In the basic type of message,
three bits are used:

ASSOCIATED DATA PRESENT (A/D) is ON if an Associated Data block
follows the Message Proper. 0 if only a message proper is present in
the network message. The value of this bit is enforced by the
network adapter firmware.

BURST MODE (BST) Enables a special mode for time critical transfers
where a single HYPERchannel A coaxial trunk is dedicated during
transmission of the network message. Not recommended for anything
that won't cause peripheral device overruns if data isn't delivered
once message transmission starts.

EXCEPTION (EXC) Indicates to some channel programmed host interfaces
that the message is "out of band" in some way and requires special
processing.

ACCESS CODE

A feature to permit adapters to share use of a cable yet still permit
an "access matrix" of which adapter boxes and physically talk to
which others. Not currently in use by anyone, support is being
discontinued.

TO ADDRESS

Consists of three parts. The high order 8-bits contains the physical
address of the network adapter box which is to receive the message.
The low order 8-bits are interpreted in different ways depending on
the nature of the receiving network adapter. If the receiving
adapter has different host "ports," then the low order bits of the TO
field are used to designate which interface is to receive the
message. On IBM data channels, the entire "logical" TO field is
interpreted as the subchannel on which the incoming data is to be
presented. Parts of the logical TO field that are not interpreted by

the network adapter are passed to the host for further
interpretation.

FROM ADDRESS

The FROM address is not physically used during the process of
transmitting a network message, but is passed through to the
receiving host so that a response can be returned to the point of
origin. In general, reversing the TO and FROM 16-bit address fields
and the TO and FROM trunk masks can reliably return a message to its
destination.

MESSAGE TYPE

The following two bytes are reserved for NSC. Users have been
encouraged to put a zero in byte 8 and anything at all in byte 9 so
as to not conflict with internal processing of messages by NSC
firmware. In the past, this field has been loosely defined as
carrying information of interest to NSC equipment carrying the
message and not as a formal protocol type field. For example, 0xFF00
in bytes 8 and 9 of the message will cause the receiving adapter to
"loop back" the message without delivering it to the attached host.

Concurrent with this document, it is NSC's intent to use both bytes 8
and 9 as a formal "protocol type" designator. Major protocols will
be assigned a unique value in byte 8 that will (among good citizens)
not duplicate a value generated by a different protocol. Minor
protocols will have 16-bit values assigned to them so that we won't
run out when 256 protocols turn up. Any interested party could
obtain a protocol number or numbers by application to NSC. In this
document, protocol types specific to IP protocols are assigned.

TO ADDRESSES AND OPEN DRIVER ARCHITECTURE

Since not all 16-bits of the TO address are used for the physical
delivery of the network message, the remainder are considered
"logical" in that their meaning is physically determined by host
computer software or (in cases such as the FIPS data channel) by
hardware in the host interface.

Since HYPERchannel is and will be used to support a large variety of
general and special purpose protocols, it is desirable that several
independent protocol servers be able to independently share the
HYPERchannel network interface. The implementation of many of NSC's
device drivers as well as those of other parties (such as Cray
Research) support this service. Each protocol server that wishes to
send or receive HYPERchannel network messages logically "connects" to
a HYPERchannel device driver by specifying the complete 16-bit TO

address it will "own" in the sense that any network message with that
TO address will be delivered to that protocol server.

The logical TO field serves a function similar to the TYPE byte in
the Ethernet 802.2 message header, but differs from it in that the
width of the logical TO field varies from host to host, and that no
values of the logical TO address are reserved for particular
protocols. On the other hand, it is possible to have several
"identical" protocols (such as two independent copies of IP with
different HYPERchannel addresses) sharing the same physical
HYPERchannel interface. This makes NSC's addressing approach
identical to the OSI concept that the protocol server to reach is
embedded within the address, rather than the IP notion of addressing
a "host" and identifying a server through a message type.

Since the HYPERchannel header also has a "message type" field, there
is some ambiguity concerning the respective roles of the message type
and logical TO fields:

  o   The logical TO field is always used to identify the protocol
       server which will receive the message. Once a server has
       specified the complete TO address for the messages it wishes to
       receive, the message will not be delivered to a different
       protocol server regardless of the contents of the message type
       field.

  o   Although the "type" field cannot change the protocol server at
       the final destination of the message, the type field can be used
       by intermediate processes on the network to process the message
       before it reaches the server destination. An obvious example is
       the 0xFF00 message loopback type function, where network
       processing to loop back the message results in nondelivery to
       the TO address. In the future, intermediate nodes may process
       "in transit" messages based on the message type only for
       purposes such as security validation, aging of certain
       datagrams, and network management.

EXTENDED (32-BIT ADDRESS) MESSAGE PROPER HEADER

   In the original days of HYPERchannel, the limitation of 256 adapter
   "boxes" that could be addressed in a network message was deemed
   sufficient as 40 or so adapters was considered a "large" network. As
   with the Ethernet, more recent networks have resulted in a need to
   address larger networks. Although a few ad hoc modes have existed to
   address larger HYPERchannel networks for some years, newer
   technologies of HYPERchannel equipment have logically extended the
   network message to support 32-bits of addressing, with 24 of those
   bits to designate a physical network adapter.

This 32-bit header has been designed so that existing network
adapters are capable of sending and receiving these messages. Only
the network bridges need the intelligence to select messages
designated for them.

```
      +-------------------------------+----------------------------+
  0   |          Trunks to Try        |         Message Flags      |
      |    TO trunks  |  FROM trunks  |GNA|CRC|   |SRC|EXC|BST|A/D|
      +--------------+----------------+---+---+--+--+---+---+---+---+
  2   |          TO Domain #          |         TO Network #       |
      |                               |                            |
      +-------------------------------+----------------------------+
  4   |O|    Physical addr of         |                 | TO Port  |
      |N|   destination adapter (TO)  |                 | number   |
      +-------------------------------+----------------------------+
  6   |O| Physical addr of source     |                 |FROM port |
      |N|     adapter (FROM)          |                 | number   |
      +-------------------------------+----------------------------+
  8   |                        Message type                        |
      |                                                            |
      +-------------------------------+----------------------------+
 10   |         FROM Domain #         |        FROM Network #      |
      |                               |                            |
      +-------------------------------+----------------------------+
 12   |          - reserved -         |          age count         |
      |                               |                            |
      +-------------------------------+----------------------------+
 14   |      Next Header Offset       |      Header End Offset      |
      |        (normally 16)          |        (normally 16)       |
      +-------------------------------+----------------------------+
 16   |               Start of user protocol                       |
      |           bytes 16 - 64 of message proper                  |
      |                                                            |
      +-------------------------------+----------------------------+


        Associated Data
      +------------------------------------------------------------+
      |                                                            |
      |     As with basic format network messages                 |
      |                                                            |
      +------------------------------------------------------------+
```

ADDRESS RECOGNITION AND MESSAGE FORWARDING

   With the 32-bit form of addressing, NSC is keeping with the premise
   that the native HYPERchannel address bears a direct relation to the
   position of the equipment in an extended HYPERchannel network.

   Each collection of "locally" attached NSC network adapters that are
   connected by coax or fiber optic cable (with the possible addition of
   nonselective repeaters such as the ATRn series) is considered a
   "network". Each network can have up to 256 directly addressable
   adapters attached to it which can be reached by the basic format

network message.

Existing bridges or "link adapters" can be programmed to become
"selective repeaters" in that they can receive network messages
containing a subset of network addresses send them over the bridge
medium (if present) and reintroduce them on the other network. Such
interconnected local area networks are considered a single network
from an addressing point of view.

A large NSC network can have up to 64K networks which can be
complexly interconnected by network bridges and/or "backbone"
networks which distribute data between other networks. To simplify
the mechanics of message forwarding, the 16-bit network field is
divided into two eight quantities, a "network number" identifying
which network is to receive the message and a "domain number" which
specifies which network of networks is the recipient.

The bridge technology adapters which move messages between networks
have address recognition hardware which examines all the 24-bits in
bytes 2-5 of the network message header to determine if the bridge
should accept the message for forwarding. At any given instant of
time in the network, each bridge will have a list of networks and
domains that it should accept for forwarding to a network at the
other end of the bridge. Each Adapter (Including Newer Technology
host adapters) contains in address recognition hardware:

  o   domainmask -- a 256-bit mask of domain numbers that should be
      accepted for forwarding (not local processing) by this adapter.
  o   MyDomain -- the value of the domain on which this host
      adapter or bridge end is installed.
  o   NetworkMask -- a 256-bit mask of network numbers that should be
      accepted for forwarding by this adapter.
  o   MyNetwork - the value of the network on which this host
      adapter or bridge end is installed.
  o   AddressMask -- A 256-bit mask of the local network addresses
      that should be accepted by the adapter.
  o   MyAddress -- the "base address" of the box, which must be
      supplied in any message that is directed to control processes
      within the adapter, such as a loopback message.

  Address recognition takes place using the algorithm:

          IF Domain IN DomainMask OR
             IF (Domain = MyDomain AND Network IN NetworkMask) OR
                IF (Domain = MyDomain AND Network = MyNetwork AND
                   Address IN AddressMask) THEN accept-message
                                        ELSE ignore-message.

   This algorithm means that an adapter's hardware address recognition
   logic will accept any messages to the box itself, any secondary or
   aliased local addresses owned by the adapter, and any message
   directed to a remote network or domain that that particular adapter
   is prepared to forward.

32-BIT MESSAGE FIELDS

TRUNK MASK

   Is as in the basic network message. Messages that are to be
   delivered outside the immediate network should have 0xFF in this byte
   so that all possible trunks in intermediate networks should be tried.
   Locally delivered 32-bit messages may still contain specially
   tailored trunk masks to satisfy local delivery needs.

MESSAGE FLAGS

   The currently defined bits remain as before. Three new bits have
   been defined since that time.

   CRC (END-END MESSAGE INTEGRITY). Newer technology host adapters are
   capable of generating a 32-bit CRC for the entire network message as
   soon as it is received over the channel or bus interface from the
   host. This 32-bit CRC is appended to the end of the associated data
   block and is preserved through the entire delivery process until it
   is checked by the host adapter that is the ultimate recipient of the
   message, which removes it. This end to end integrity checking is
   designed to provide a high degree of assurance that data has been
   correctly moved through all intermediate LAN's, geographic links, and
   internal adapter hardware and processes.

   SRC (SOURCE FROM ADDRESS CORRECT). This bit is provided to take
   advantage of the physical nature of the network address to optionally
   verify that the 32-bit FROM address provided in the network message
   is in fact the location that the message originated. If the bit is
   not set by the transmitting host, no particular processing occurs on
   the message. If the bit is set, then all intermediate adapters
   involved in the delivery of the message have the privilege of turning
   the bit off if the received message FROM address is not a TO address
   that would be delivered to the originator if the message were going
   the opposite direction.

   If the message is received by a host computer with this bit still
   set, then the FROM address is guaranteed correct in the sense that
   returning a message with TO and FROM information reversed will result
   in delivery of the message to the process that actually originated

   it. By careful attention to the physical security of adapters and
   intermediate links between networks, a high degree of security can be
   built into systems that simply examine the FROM address of a message
   to determine the legitimacy of its associated request.

   GNA (GLOBAL NETWORK ADDRESSING). This bit ON indicates that 32-bit
   addressing is present in the message. When this bit is on, bytes 2-3
   (Domain and Network numbers) should also be nonzero.

TO ADDRESS

   Four bytes contain the TO address, which is used to deliver the
   network message as described in "Address Recognition and Message
   Forwarding" on page 8. The "logical" part of the TO address is used
   to designate a protocol server exactly as in the basic format network
   message header.

   The existing "address" field has its high order bit reserved as an
   outnet bit for compatibility with existing A-series network adapter
   equipment. Were it not for this bit, the A-series adapters would
   attempt to accept messages that were "passing through" the local
   network on their way elsewhere simply because the address field
   matched while the the Domain and Network numbers (ignored by the A-
   series adapters) were quite different.

   This "outnet" bit is used in the following way:

    o   All network adapters (of any type) in an extended set of
         networks containing A-Series adapters that will ever use 32-bit
         addressing must have their addresses in the range 00-7F (hex.)

    o   If a message is to be sent to a destination on a nonlocal
         network and domain on such an extended network, then the
         high order bit of the address field is turned on.

    o   When the last bridge in the chain realizes that it is about to
         forward the message to its final destination (the Domain and
         Network numbers are local), then it turns the Outnet bit off.
         This will result in local delivery to the destination adapter.

FROM ADDRESS

   The FROM address follows the same logic as the TO address in that any
   message can be returned to its source by reversing the FROM and TO
   fields of the message. Since so many protocols examine byte 8 of the
   message to determine its type, the FROM field has been split so that
   the Domain and Network numbers extend into bytes 10-11.

MESSAGE TYPE

   This field (informally defined in the past) has been extended to 16-
   bits so that a unique value can be assigned to any present or future
   protocol which is layer on HYPERchannel messages for either private
   or public use.

AGE COUNT

   This field serves the same purpose as the IP "time to live" in that
   it prevents datagrams from endlessly circulating about in an
   improperly configured network. Each time a 32-bit message passes
   through a bridge, the Age Count is decremented by one. When the
   result is zero, the message is discarded by the bridge.

NEXT HEADER OFFSET AND HEADER END OFFSET

   These are used as fields to optionally provide "loose source
   routing", where a list of 32-bit TO addresses can be provided by the
   transmitter to explicitly determine the path of a message through the
   network. If this feature is not used, both these fields would
   contain the value 16 (decimal) to both indicate extra TO addresses
   are absent and that the beginning of protocol data following the
   HYPERchannel header is in byte 16.

   Although it is conceivable that a HYPERchannel IP process could use
   this source routing capability to direct messages to hosts or
   gateways, this capability is not felt to be of sufficient value to IP
   to build it into a HYPERchannel IP protocol.

   In the future, all higher level protocols should be able to examine
   Header End Offset to determine the start of the higher level protocol
   information.

BROADCASTING

   NSC message forwarding protocols use low level link protocols to
   negotiate transmission of a message to its next destination on the
   network. Furthermore, NSC network boxes often "fan out" so that
   several hosts share the same network transmission equipment as in the
   A400 adapter. Both these characteristics mean that providing a
   genuine broadcast capability is not a trivial task, and in fact no
   current implementations of NSC technology support a broadcast
   capability.

   The last several years have seen broadcast applications mature to the
   point where they have virtually unquestioned utility on a local and
   sometimes campuswide basis. Accordingly, new NSC technologies will

support a broadcast capability. Information on the use of this
capability is included here as it is essential to the discussion of
the Address Resolution Protocol later in this document.

Broadcast capability will be supported only with the extended (32-bit
address) message format. A broadcast message will have the following
general appearance:

```
 byte   Message Proper
       +----------------------------+----------------------------+
   0   |        Trunks to Try       |         Message Flags      |
       |   TO trunks  |  FROM trunks |GNA|CRC|    |SRC|EXC|BST|A/D|
       +-------------+--------------+---+---+--+--+---+---+---+---+
   2   |      TO Domain Number      |      TO Network Number     |
       |         or 0xFF            |         or 0xFF            |
       +----------------------------+----------------------------+
   4   |           0xFF             |  Broadcast channel number  |
       |                            |                            |
       +----------------------------+----------------------------+
   6   |O| Physical addr of source  |                 |FROM port |
       |N|     adapter (FROM)       |                 |  number  |
       +----------------------------+----------------------------+
   8   |                    Message type                         |
       |                                                         |
       +----------------------------+----------------------------+
  10   |     FROM Domain Number     |     FROM Network Number    |
       |                            |                            |
       +----------------------------+----------------------------+
  12   |        - reserved -        |         age count          |
       |                            |                            |
       +----------------------------+----------------------------+
  14   |     Next Header Offset      |     Header End Offset      |
       |        (normally 16)        |        (normally 16)       |
       +----------------------------+----------------------------+
  16   |                 Start of user protocol                  |
       |            bytes 16 - 64 of message proper              |
       |                                                         |
       +----------------------------+----------------------------+

       Associated Data
     +-----------------------------------------------------------+
     |                                                           |
     |    As with basic format network messages                 |
     |    Maximum associated data size 1K bytes.                 |
     |                                                           |
     +-----------------------------------------------------------+
```

TRUNKS TO TRY AND MESSAGE FLAGS

   These fields are defined just as with a normal 32-bit message. All
   bits in the Message Flags field are valid with broadcast modes.

BROADCAST ADDRESS

   For Domain, Network and Adapter Address fields, the value 0xFF is
   reserved for use by the broadcast mechanism. A value of 0xFF in the
   adapter address field indicates to the local network hardware that
   this message is to be sent to all connected network equipment on the
   individual network.

   A value of 0xFF in the network or domain fields, respectively
   indicates a request that the scope of the broadcast exceed the local
   network. The bridging link adapters will receive the broadcast
   message along with everyone else and will examine the "Broadcast
   Channel" field and their internal switches to determine if the
   message should be forwarded to other remote networks.

   If the Network and Domain fields contain the local network and
   domain, then the broadcast message will only be broadcast within the
   local network. If a remote Network and Domain is specified, then the
   message will be delivered as a single message to the remote network
   and broadcast there.

BROADCAST CHANNEL

   Since individual hosts and protocol servers generally are not
   interested in all broadcast messages that float about the network, a
   filtering mechanism is provided in the header and network adapter
   equipment so that only proper classes of broadcast messages are
   delivered to the end point.

   Broadcast channel numbers in the range 00-0xFF will be assigned by
   NSC much like the "message type" field. Host protocol servers
   specify a specific TO address containing a channel number (such as
   0xFF04) when they bind themselves to the HYPERchannel device driver.
   The driver and the underlying equipment will deliver only broadcast
   messages with the correct channel number to the protocol server. If
   a protocol server wishes to receive several different broadcast
   messages, it must bind itself to the driver several times with the
   desired addresses.

   Link adapters that are prepared to handle multinetwork broadcast
   messages may be equipped with switches to determine which broadcast
   channels will be propagated into the next network. Since
   multinetwork broadcast is an arrangement that must be configured with

   care, these switches are off by default.

FROM ADDRESS

   The FROM address is constructed just as with a normal 32-bit network
   message. The Source Address Correct bit is processed just as with a
   normal message.

MESSAGE TYPE

   Message type is defined as with normal messages. Presumably
   broadcast applications will have unique message types that are not
   generally found in normal messages.

AGE COUNT

   Age count is vitally important in a multinetwork broadcast as "loops"
   in the network can cause a great deal of activity until all the
   progeny of the original broadcast message die out.

PROTOCOL SPECIFICATION

   This section contains information on the technique used to
   encapsulate IP datagrams on the HYPERchannel network message. It
   contains three sections to describe three protocol packagings:

    o   The technique used to encapsulate IP datagrams on the basic
        16-bit network message. This is a de facto standard that has
        been in use for several years and is documented here to make it
        official.

    o   The encapsulation technique for IP datagrams on 32 bit network
        messages.

    o   The definition of an Address Resolution Protocol on
        HYPERchannel.

BASIC (16-BIT) MESSAGE ENCAPSULATION
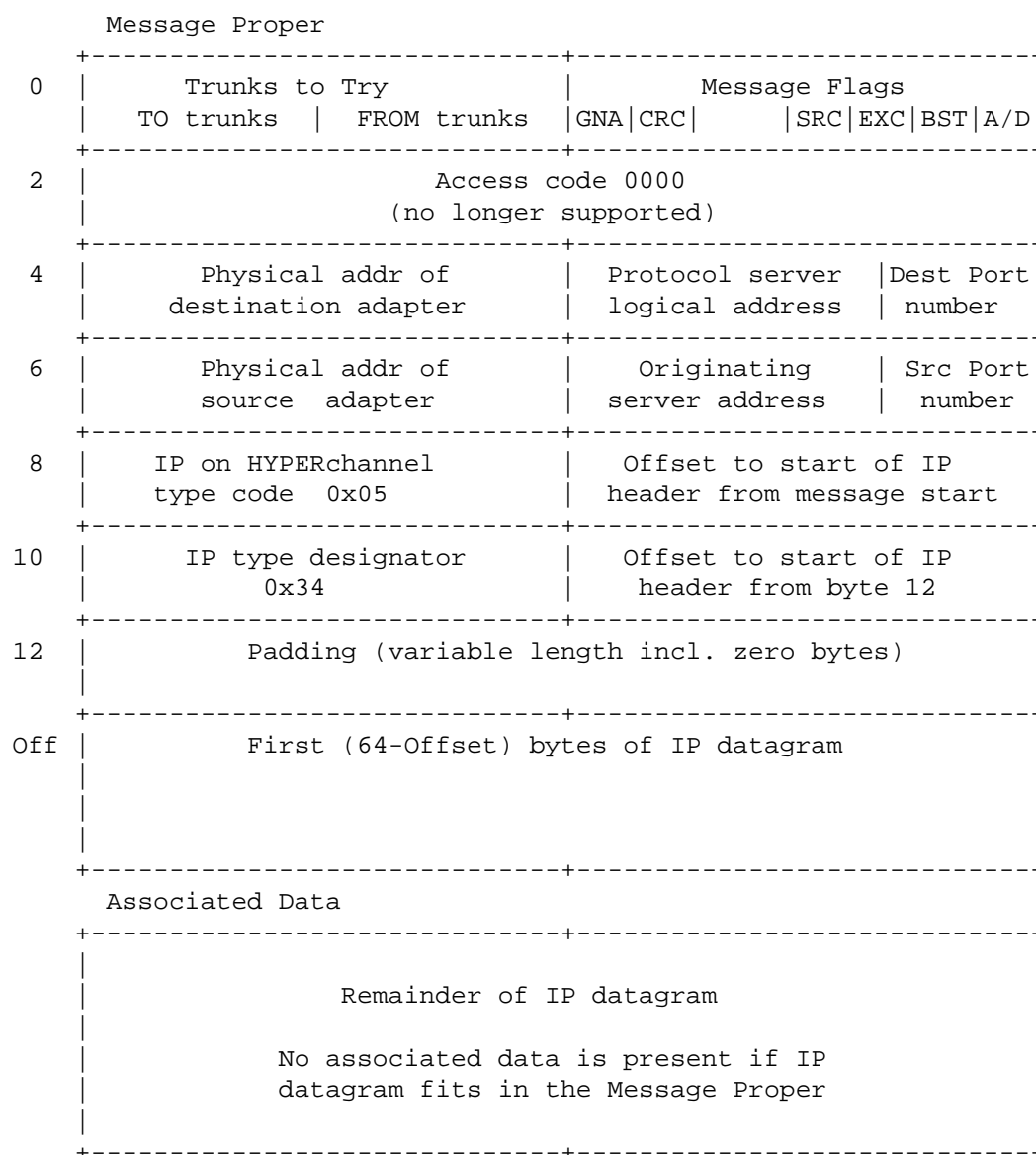
```
         Message Proper
         +----------------------------+----------------------------+
    0    |         Trunks to Try      |         Message Flags      |
         |  TO trunks  |  FROM trunks |GNA|CRC|    |SRC|EXC|BST|A/D|
         +----------------------------+----------------------------+
    2    |                    Access code 0000                     |
         |                    (no longer supported)                |
         +----------------------------+----------------------------+
    4    |        Physical addr of    | Protocol server  |Dest Port|
         |        destination adapter | logical address  | number  |
         +----------------------------+----------------------------+
    6    |        Physical addr of    |    Originating    | Src Port|
         |        source  adapter     | server address    | number  |
         +----------------------------+----------------------------+
    8    |    IP on HYPERchannel      |    Offset to start of IP    |
         |    type code   0x05        |    header from message start|
         +----------------------------+----------------------------+
   10    |    IP type designator      |    Offset to start of IP    |
         |         0x34               |     header from byte 12     |
         +----------------------------+----------------------------+
   12    |        Padding (variable length incl. zero bytes)       |
         |                                                         |
         +----------------------------+----------------------------+
  Off    |       First (64-Offset) bytes of IP datagram            |
         |                                                         |
         |                                                         |
         |                                                         |
         +----------------------------+----------------------------+
         Associated Data
         +----------------------------+----------------------------+
         |                                                         |
         |                                                         |
         |             Remainder of IP datagram                    |
         |                                                         |
         |          No associated data is present if IP            |
         |          datagram fits in the Message Proper            |
         |                                                         |
         +----------------------------+----------------------------+
```

TRUNK MASK

   From the vantage of an IP driver, any trunk mask is valid so long as
   it results in successful delivery of the HYPERchannel network message
   to its destination. There is no reason to check this field for
   validity on reception of the message. Specification of the Trunk
   Mask on output is a local affair that could be specified by the
   transmitting driver's address resolution tables.

MESSAGE FLAGS

   No use is made of the Flags field (byte 1) other than to
   appropriately set the Associated Data bit. Burst Mode and the
   Exception bit should not be used with IP.

ACCESS CODE

   Although some current implementations of IP on HYPERchannel support
   the access code, no one appears to be using it at the current time.
   Since this field is currently reserved for the use of 32-bit
   addresses, no value other than 0000 should be placed in this field.

TO ADDRESS

   The TO field is generally obtained by a local IP driver through a
   table lookup algorithm where a 16-bit TO address is found that
   corresponds to the IP address of a local host or gateway. The high
   order bits of the TO address of course refer to the adapter number
   the adapter attached to the destination host.

   The logical TO field should contain the protocol server address of
   the HYPERchannel IP driver for that host as determined by the host's
   system administrator. Many HYPERchannel TCP/IP drivers in the field
   today are not "open" in that any network message delivered to that
   host will be presumed to be an IP datagram regardless of the logical
   TO field; however any transmitting IP process should be capable of
   generating the entire 16-bit TO field in order to generate a message
   capable of reaching a destination IP process.

   The process of determining which HYPERchannel address will receive an
   IP datagram based on its IP address is a major topic that is covered
   in "Address Resolution".

FROM ADDRESS

   The FROM address is filled in with the address that the local driver
   expects to receive from the network, but no particular use is make of
   the FROM address.

MESSAGE TYPE

   Network Systems requests that a value of 5 (decimal) be placed in
   this byte to uniquely indicate that the network message is being used
   to carry IP traffic. No other well-behaved protocol using
   HYPERchannel should duplicate this value of 5.

   Many current implementations of IP on HYPERchannel place a zero or
   other values in this field simply because no value was reserved for
   IP usage. Transmitting versions of IP should always place a 5 in
   this field; receiving IP's should presume a delivered message to be
   an IP datagram until proven otherwise regardless of the contents of
   the Message Type field.

   Developers should note that it is often convenient to permit
   reception of the value 0xFF00 in bytes 8 and 9 of the IP datagram.
   Transmitting a message with this value will cause it to be looped
   back at the destination adapter and returned to the protocol server
   designate in the FROM address. This permits the developer have host
   applications talk to others on the same host for purposes of network
   interface or other protocol debugging.
IP HEADER OFFSET

   Byte 9 contains the offset to the start of the IP header within the
   message proper, such that the Message Proper address plus the IP
   header offset generates the address of the first byte of the IP
   header (at least on byte addressable machines.)

   This field is redundant with the offset field in byte 11, and is
   present for cosmetic compatibility with 32-bit implementations. On
   reception, the value in byte 11 should take precedence.

   As part of the migration to larger HYPERchannel headers, this field
   will become significant with the 32-bit addressing format, as the
   length of the header is no longer 10 bytes and byte 11 is used for
   other purposes.

IP TYPE DESIGNATOR

   Early implementations of IP drivers on HYPERchannel wanted to leave
   bytes 8 and 9 alone for NSC use and place a "message type" field in
   later in the message. A value of 0x34 had been selected by earlier
   developers for reasons that are now of only historical interest.
   Once again, implementations should generate this value on
   transmission, but not check it on input, assuming that an IP datagram
   is present in the message.

IP HEADER OFFSET

   This value is used by a number of commercial implementations of IP on
   HYPERchannel to align the start of the IP header within the network
   message. This offset is relative to byte 12 of the network message
   so that a value of zero indicates that the IP header begins in byte
   12. This value should be both correctly generated on transmission,
   and always respected on input processing.

The maximum permissible offset in this field is 52 indicating that
the IP header begins at the start of the associated data block.

IP DATAGRAM CONTENTS

Beginning at the offset designated in byte 11, the IP datagram is
treated as a contiguous block of data that flows from byte 63 of the
message proper into the first byte of associated data, so that the
entire message plus data is treated as a single contiguous block.

If the IP header is small enough to fit within the entire network
message, then only the message proper is transmitted. The length of
the message proper sent should always be 64 bytes, even if the IP
datagram and HYPERchannel header do not occupy all 64 bytes of the
message proper.

If the datagram flows over into the associated data, then both
message and data are sent. Since a number of machines cannot send a
length of data to the HYPERchannel that is an exact number of bytes
(due to 16-64 bits on the channel bus,) the length of the associated
data received should not be used as a guide to the length of the IP
datagram -- this should be extracted from the IP header. A driver
should verify, of course, that the associated data received is at
least as long as is needed to hold the entire IP datagram.

COMPATIBILITY WITH EXISTING IMPLEMENTATIONS

The basic format described here is clearly a compromise between
several implementations of IP on HYPERchannel. Not all existing
implementations are interoperable with the standard described above.
Currently there are two known "families" of IP HYPERchannel drivers
in existence:

THE "CRAY-NASA AMES" PROTOCOL

This protocol is in the widest production use and has the largest
number of supported drivers in existence. It is interoperable and
identical with the standard described above with the sole exception
that bytes 8 and 9 are set to zero by these drivers. As these bytes
are ignored by most implementations of this driver, they have been
assigned values to formalize the use of the message type field and to
make it consistent with the 32-bit protocol.

THE "TEKTRONIX-BERKELEY" PROTOCOL

This protocol was historically the first IP on HYPERchannel
implementation developed (at Tektronix) and subsequently made its way
to Berkeley and BSD UNIX. This protocol is not interoperable with

the standard described above due to several distinct differences.

First, bytes 8 through 11 are always zero. The IP header always
starts on byte 12. Comments in some of these drivers designate byte
11 as an "IP header offset" field, but apparently this value is never
processed.

The major difference (and the incompatibility) concerns the packaging
of the IP datagram into the network message. Due to historical
difficulties in the early 80's with the sending and receiving of very
small blocks of associated data on VAXes, this protocol the takes a
curious approach to the placement of the IP header and the headers of
higher level protocols (such as TCP or UDP.)

  o   If the entire length of the IP datagram is 54 bytes or less,
       it is possible to fit the entire datagram and the HYPERchannel
       header in the 64 byte message proper. In this case, no
       associated data is sent; only a message proper is used to carry
       the data. The length of the message proper transmitted is the
       exact length needed to enclose the IP datagram; no padding bytes
       are sent at the end of the message.

  o   If the length of the IP header is greater than 54 bytes, then:

       -    All higher level protocol information (TCP/UDP header and
            their associated data fields) are placed in the associated
            data block, with the TCP/UDP header beginning at the start
            of the associated data block.

       -    On transmission, the length of the message proper
            transmitted is set to the length of the HYPERchannel header
            plus the IP header -- it is not padded out to 64 bytes.
            The length of the associated data sent should be sufficient
            to accommodate the TCP/UDP header and its data fields.

WHICH PROTOCOL IS BEST?

   In choosing which to follow, the "Cray-Ames" approach was taken for
   several reasons:

   1. Cray Research has performed exemplary work in dealing with other
       vendors to provide IP on HYPERchannel from the Cray computers to
       other hosts. As a result, there are 4 or 5 vendor supported
       implementations of IP on HYPERchannel that use this approach.

   2. The two part structure of the message proper has its uses when a
       machine wishes to make protocol decisions before staging the
       transfer of an immense block of associated data into memory.
       Many network coprocessors and intelligent I/O subsystems find it
       simpler to read in the entire network message before deciding
       what to do with it. Arbitrarily catenating the two components
       does this best and permits streaming of messages from future
       technology network adapters.

   3. Some TCP users (mostly secure DoD sites) intend to load up IP
       datagrams with optional fields in the future. The
       Tektronix-Berkeley implementation has problems if the IP header
       length exceeds 54 bytes.

EXTENDED (32-BIT) MESSAGE ENCAPSULATION

```
        Message Proper
        +-----------------------------+-----------------------------+
  0     |        Trunks to Try        |1|        Message Flags       |
        |   TO trunks  |  FROM trunks |GNA|CRC|     |SRC|EXC|BST|A/D|
        +-----------------------------+-----------------------------+
  2     |      Destination  Domain    |     Destination  Network    |
        |          Number             |            Number           |
        +-----------------------------+-----------------------------+
  4     |O|      Physical addr of     |   Protocol server |Dest Port|
        |N|    destination adapter    |   logical address |  number |
        +-----------------------------+-----------------------------+
  6     |O|      Physical addr of     |      Originating   | Src Port|
        |N|      source  adapter      |   server address   |  number |
        +-----------------------------+-----------------------------+
  8     |      IP on HYPERchannel     |     Offset to start of IP    |
        |      type code   0x06       |        datagram header       |
        +-----------------------------+-----------------------------+
 10     |    Source Domain Number     |    Source Network Number     |
        |                             |                              |
        +-----------------------------+-----------------------------+
 12     |        - reserved -         |          Age Count           |
        +-----------------------------+-----------------------------+
 14     |      Next Header Offset      |      Header End Offset       |
        |                             |         (usually 16)         |
        +-----------------------------+-----------------------------+
 16     |        Padding to IP header start (usually 0 bytes)        |
        |                                                            |
        +-----------------------------+-----------------------------+
Off|        Entire IP datagram if datagram length <= (64-Offset)    |
        |                                                            |
        |        else first (64-Offset) bytes of IP datagram         |
        +-----------------------------+-----------------------------+


        Associated Data
        +-----------------------------+-----------------------------+
        |                                                            |
        |                 Remainder of IP datagram                   |
        |                                                            |
        |           No associated data is present if IP              |
        |           datagram fits in the Message Proper              |
        |                                                            |
        +-----------------------------+-----------------------------+
```

TRUNK MASK

   From the vantage of an IP driver, any trunk mask is valid so long as

it results in successful delivery of the HYPERchannel network message
to its destination. There is no reason to check this field for
validity on reception of the message. Specification of the Trunk
Mask on output is a local affair that can be specified by the
transmitting driver's address resolution tables.

The use of 0xFF in this value is strongly encouraged for any message
other than those using exotic trunk configurations on a single local
network.

MESSAGE FLAGS

Several new bits have been defined here.

EXTENDED ADDRESSING. This bit should be set ON whenever a 32-bit
address (Network and/or Domain numbers nonzero) is present in the
message. It should always be OFF with the 16-bit message header. If
this bit is improperly set, delivery of the message to the (apparent)
destination is unlikely.

END-TO-END CRC. Some newer technology adapters are equipped to place
a 32-bit CRC of the associated data at the end of the associated data
block when this bit is on. Similarly equipped adapters will examine
the trailing 32-bits of associated data (when the bit is on) to
determine if the message contents have been corrupted at any stage of
the transmission.

Transmitting device drivers should include the ability to set this
bit on transmission as a configuration option similar to the specific
HYPERchannel device interface used. The bit should be generated to
be turned ON if the HYPERchannel IP driver is attached to an adapter
equipped to generated CRC information -- it should be left OFF in all
other circumstances.

If a message arrives at the host with the CRC bit still on, this
indicates that the CRC information was placed at the end of
associated data by the transmitting adapter and not removed by the
receiving adapter; thus the associated data will be four bytes longer
than otherwise expected. Since the IP datagram length is self
contained in the network message, this should not impact IP drivers.

It is possible for host computers to both generate and check this CRC
information to match the hardware assisted generation and checking
logic in newer network adapters. Contact NSC if there are particular
applications requiring exceptional data integrity that could benefit
from host generation and checking.

FROM ADDRESS CORRECT. This bit should be set by all transmitting IP
drivers who have endeavored to provide a completely correct FROM
address that properly reflects the adapter interface used. No action
should be taken on this bit by the receiving IP driver at this time.
Additional work needs to be done to determine the action an IP driver
should take if it detects a real or imagined "security violation"
should a message arrive with this bit absent.

TO ADDRESS

The TO address logically constitutes bytes 2-5 of the network
message.

NETWORK AND DOMAIN NUMBERS. The Network and Domain numbers should
both be nonzero when 32-bit addressing is used. If the message is
local in nature, then the local Network and Domain numbers should be
placed in this field.

ADAPTER ADDRESS. Contains the adapter address as in the basic
message. The high order bit of this eight bit field (the "outnet"
bit) should be set to zero if the destination network and domain are
the same as the transmitting host's. The high order bit should be
set to one if the destination host is not in the local network or
domain.

LOGICAL TO AND SUBADDRESS. The logical TO field should contain the
protocol server address of the HYPERchannel IP driver for that host
as determined by the host's system administrator.

FROM ADDRESS

The FROM address is filled in with the address that the local driver
expects to receive from the network, but no particular use is made of
the FROM address.

MESSAGE TYPE

The value 6 must be placed in this byte to uniquely indicate that the
network message is being used to carry IP traffic. No other well-
behaved protocol using HYPERchannel should duplicate this value of 6.

Note that all IP drivers should be prepared to send and receive the
basic format network messages using the 16-bit HYPERchannel
addresses. The driver can distinguish an incoming network message by
the value of byte 8 -- 32-bit messages will always have a 6 in byte
8, while 16-bit messages should have a 5 here. For interoperability
with older drivers, a value of 0 here should be treated as 16 address
bit messages.

IP HEADER OFFSET

   Byte 9 contains the offset to the start of the IP header within the
   message proper, such that the Message Proper address plus the IP
   header offset generates the address of the first byte of the IP
   header (at least on byte addressable machines.)

   Unlike the 16-bit header, receiving IP drivers should assume that
   this field contains a correct offset to the IP header and examine the
   information at that offset for conformance to an IP datagram header.

   Valid offsets are in the range of 16 through 44 bytes, inclusive.
   The limitation of 44 bytes is imposed so that routing decisions on
   the vast majority of IP datagrams can be made by examining only the
   message proper, as the basic IP datagram will fit into the message
   proper if it begins at an offset of 44.

IP DATAGRAM CONTENTS

   The message and data are treated as logically contiguous entities
   where the first byte of associated data immediately follows the 64th
   byte of the message proper.

   If the entire IP datagram is less than or equal to (64-offset) bytes
   in length it will fit into the Message Proper. If so, only a message
   proper containing the HYPERchannel header and IP datagram is sent on
   the network.

   If the IP datagram is greater than this length, the IP datagram
   spills over into the associated data. On transmission, a 64 byte
   message proper is sent followed by as many bytes of associated data
   as are needed to send the entire datagram.

   On reception, the message proper can be read into the start of an IP
   input buffer and the associated data read into memory 64 bytes from
   the start of the message. If the received message is in fact a 32-
   bit address message, no "shuffling" of the message will be required
   to build a contiguous IP datagram -- it's right there at buffer+16.

ADDRESS RESOLUTION PROTOCOL

   Address Resolution Protocol has achieved a great deal of success on
   the Ethernet as it permits a local IP network to configure itself
   simply by having each node know its own IP address. Those unfamiliar
   with the intent, protocol, and logic of the Address Resolution
   Protocol should refer to RFC-826, "An Ethernet Address Resolution
   Protocol" [2].

A later section of this document describes four techniques where a
target HYPERchannel address is to obtained given the destination's IP
address. The protocol is defined in this section for completeness.

```
         Message Proper
      +----------------------------+----------------------------+
   0  |        Trunks to Try       |1|        Message Flags     |
      |  TO trunks  |  FROM trunks |GNA|CRC|     |SRC|EXC|BST|A/D|
      +----------------------------+----------------------------+
   2  |       Server Domain or     |     Server Network or      |
      |          0xFF              |          0xFF              |
      +----------------------------+----------------------------+
   4  |  Server Adapter Address or | Server logical addr/port or|
      |          0xFF              |          07                |
      +----------------------------+----------------------------+
   6  |O|     Physical addr of     |    Originating   | Src Port|
      |N|     source  adapter      |  server address  |  number |
      +----------------------------+----------------------------+
   8  |               NSC ARP type code                         |
      |          07                |          00                |
      +----------------------------+----------------------------+
  10  |       Source Domain        |      Source Network        |
      +----------------------------+----------------------------+
  12  |       - reserved -         |        Age Count           |
      +----------------------------+----------------------------+
  14  |     Next Header Offset     |     Header End Offset       |
      |       (usually 16)         |       (usually 16)         |
      +----------------------------+----------------------------+
  16  |      Padding to start of IP info (usually 0 bytes)      |
      +----------------------------+----------------------------+
```

```
           +-----------------------------+-----------------------------+
      Off  |          ARP hardware address type for HYPERchannel      |
           |                             8                             |
           +-----------------------------+-----------------------------+
       +2  |                   HYPERchannel protocol type             |
           |             06                             00             |
           +-----------------------------+-----------------------------+
       +4  | HYPERchannel address length |      IP address length      |
           |             6               |             4               |
           +-----------------------------+-----------------------------+
       +6  |            ARP opcode (request or reply)                 |
           +-----------------------------+-----------------------------+
       +8  |          Domain             |          Network            |
           +-          Sender's 32-bit HYPERchannel address          -+
      +10  |      Adapter address        |      Logical addr/port      |
           +-----------------------------+-----------------------------+
      +12  |                      Source's MTU size                   |
           +-----------------------------+-----------------------------+
      +14  |                             |                             |
           +-              Sender's 32-bit IP address                -+
      +16  |                             |                             |
           +-----------------------------+-----------------------------+
      +18  |          Domain             |          Network            |
           +-        Destination's 32-bit HYPERchannel address       -+
      +20  |               (to be determined on request)              |
           |      Adapter address        |      Logical addr/port      |
           +-----------------------------+-----------------------------+
      +22  |                   Destination's MTU size                 |
           |               (to be determined on request)              |
           +-----------------------------+-----------------------------+
      +24  |                             |                             |
           +-            Destination's 32-bit IP address             -+
      +26  |                             |                             |
           +-----------------------------+-----------------------------+
```

   Layout of the fields of this ARP message is a fairly straightforward
   exercise given the standards of ARP and the 32-bit message header. A
   few fields are worth remarking upon:

TO ADDRESS

   The TO address of an ARP message will be one of two classes of
   address. A "normal" address indicates that the message is an ARP
   response, or that it is an ARP request directed at an ARP server at a
   well known address on the local network. For those HYPERchannel
   networks which are equipped to broadcast, a value of 0xFFFFFF07 in
   the TO address will (by convention) be picked up only by those
   protocol servers prepared to interpret and respond to ARP messages.

The issue of which address to use in an ARP request is discussed in
the Address Resolution section.

FROM ADDRESS

   Must be the correct FROM address of the user protocol server issuing
   an ARP request. The Source Correct bit in the Message Flags byte
   should be set by this requesting server, as some ARP servers may
   someday choose to issue ARP information on an "need to know" basis in
   secure environments. With an ARP response, the FROM address will
   contain the "normal" HYPERchannel address of the protocol server
   responding to the ARP address, even if that server was reached via
   broadcast mechanisms.

   ARP responses are returned to the party specified in the FROM address
   specified in the message header, rather than the address in the
   "Source HYPERchannel Address" field within the body of the ARP
   message.

MESSAGE TYPE

   The 16-bit value 0x0700 is reserved for the exclusive use of ARP.
   Unlike IP messages, no provision is made for the ARP message to begin
   at an arbitrary offset within the message proper, so the value in
   byte 9 is an extension of the message type.

HEADER END OFFSET

   ARP uses the 32-bit addressing convention that byte 15 contains the
   offset to the start of user protocol (and hence the end of user
   protocol information). Note that this is not a substitute for the IP
   offset fields, as this field also serves as the end of HYPERchannel
   header information -- future NSC message processing code may well
   take exception to "garbage" between the actual header end and the
   start of user data.

HYPERCHANNEL HARDWARE TYPE CODE

   This 16-bit number is assigned a formal ARP hardware type of 8.

HYPERCHANNEL PROTOCOL TYPE

   On the Ethernet, this field is used to distinguish IP from all other
   protocols that may require address resolution. To be logically
   consistent, this field is identical to bytes 8 and 9 0x0600 in a 32-
   bit address HYPERchannel message carrying an IP datagram.

HYPERCHANNEL ADDRESS LENGTH

   This contains the value 6, a sufficient number of bytes to
   accommodate the four byte HYPERchannel address and 2 bytes to
   indicate the largest IP datagram size that source and destination can
   handle.

SOURCE AND DESTINATION HYPERCHANNEL ADDRESS

   This field contains the Domain, Network, and Adapter/port address of
   source and destination, respectively. A value of 0000 in the Domain
   and Network fields has special significance as this is interpreted as
   a request to send and receive 16-bit HYPERchannel headers rather than
   32-bit headers. If 32-bit headers are to be used within a single
   HYPERchannel network, then the local domain and network numbers may
   be specified.

MAXIMUM TRANSMISSION UNIT

   HYPERchannel LAN technology is such that messages of unlimited length
   may be sent between hosts. Since host throughput on a network is
   generally limited by the rate the network equipment can be
   functioned, larger transmission sizes result in higher bulk transfer
   performance. Since not every host will be able to handle the maximum
   size IP datagram, a more flexible means of MTU (maximum transmission
   unit) size negotiation than simply wiring the same value into every
   network host is needed. With this field, each host declares the
   maximum IP datagram size (not the associated data block size) it is
   prepared to receive. Transmitting IP drivers should be prepared to
   send the minimum of the source and destination IP sizes negotiated at
   ARP time.

   The MTU size sent refers to the maximum size of IP header + data. It
   does not include the length of the HYPERchannel Hardware header or
   any offset between the header and the start of the IP datagram.
   Since it is the option of the transmitting hosts to use an offset of
   up to 44 bytes a receiving host must in any event be prepared to
   receive a 64 byte Message Proper and an Associated Data block of
   MTU-20 (that is 64 - 44, or the length of the basic IP header).

        An example of a typical 16-bit packet is:

           12 bytes hardware header.
           12 bytes offset.
           40 bytes IP/TCP header.
         4096 bytes of data.
      This gives an MTU of 4136.

An example of a typical 32-bit packet is:

```
    16 bytes hardware header.
     8 bytes offset.
    40 bytes  IP/TCP header.
  4096 bytes of associated data,
This also gives an MTU of 4136.
```

The offset values are chosen so that the typical packet causes user
data to be page aligned at the start of the associated data area.
This is an implementation decision, which can certainly be modified
as required.

The maximum maximum transmission unit is 65536, the current largest
size IP datagram. In order to allow this value to fit into a 16-bit
field, the offset length is not included in the MTU. This MTU size
is not a requirement that a local host be equipped to send or receive
datagrams of that size; it simply indicates the maximum capacity of
the receiving host.

A note on trunk masks:

There is no field for specifying trunk masks. This is intentional,
as new NSC hardware will contain trunk reachability information,
eliminating the need for the host to maintain hardware configuration
layouts. All HYPERchannel messages generated as a result of an ARP
response should use 0xFF in the trunk mask.

ADDRESS RESOLUTION

This section describes techniques used by an IP driver to determine
the HYPERchannel address and header that a message should contain
given an IP datagram containing an IP address. It describes
techniques that are local to specific hosts (and hence can be
modified without regard to the activities or techniques of other
hosts) as well as techniques to use the Address Resolution Protocol
on existing HYPERchannel equipment to better manage IP addresses.

It also discusses the migration of name resolution on one of four
steps.

  1. Truncation of the IP address to form a HYPERchannel address.

  2. Local resolution of HYPERchannel addresses through configuration
      files.

  3. Centralized resolution of HYPERchannel addresses through an "ARP
      server" driven by a configuration file.

     4. Distributed resolution of HYPERchannel addresses using a "real"
         address Resolution Protocol on future HYPERchannel media
         supporting a broadcast mode.

IP ADDRESS TRUNCATION

   A number of IP on HYPERchannel implementations support modes where
   the HYPERchannel address is generated by placing the low order 16-
   bits of the IP address in the TO address of the message proper. This
   more or less treats a set of HYPERchannel boxes addressable through
   16-bit HYPERchannel addresses as a Class B IP network.

   This approach certainly offers simplicity: IP addresses are simply
   chosen to match HYPERchannel addresses and no IP address
   "configuration files" need be kept. Although this approach works in
   an environment where the HYPERchannel completely constitutes a Class
   B network, or where connection to a larger IP network is not a
   concern, its long term use is discouraged for several reasons:

    o   It simply will not work with any Class C address (the physical
        TO address is not controllable) or a Class A address (where host
        addresses are generally carefully administered.) In addition,
        it will not support subnetworks. It is quite incompatible with
        32-bit HYPERchannel addresses.

    o   By decoupling the IP and HYPERchannel addresses through more
        complex address resolution, the characters of the two addresses
        allow greater site flexibility: the IP address becomes
        "logical" in character so that an address can move about a site
        with the user or host; the HYPERchannel address maintains its
        physical character so that a HYPERchannel address carefully
        identifies the physical location of the source and destination
        within the extended HYPERchannel network.

LOCAL ADDRESS RESOLUTION

   The current state of address resolution art with IP on HYPERchannel
   works as follows: given an arbitrary IP address, the IP HYPERchannel
   driver looks up an entry with that address in a (generally hashed)
   table. If found, the table entry contains the first 6 bytes of the
   HYPERchannel header that is used to send the IP datagram to the next
   IP node on the internet. Since implementations such as the 4.3BSD
   UNIX IP are clever enough to provide its lower level drivers with the
   IP address of the next gateway as well as the destination address on
   the internet (assuming the message is not delivered locally on the
   HYPERchannel,) the number of entries in this table is more or less
   manageable, as it must only contain the IP hosts and gateway
   addresses that are directly accessible on the HYPERchannel.

CONFIGURATION FILE FORMAT

   So long as this technique of address resolution is used, the
   techniques used are exclusively local to the host in the sense that
   the techniques used to generate and store the information in the
   table are irrelevant to other hosts.

   Shown here is a typical file format. This file should probably be
   program generated from a database, as asymmetric trunk configurations
   and multiply homed hosts can cause differences in physical routing
   and trunk usage. This format is documented here to illustrate what
   sort of information must be kept at the link layer.

   The file consists of source lines each with the form:

       <type> <hostname> <trunks/flags> <domain/net> <addr> <MTU>

   an example:

           <type> <hostname>   <t/f> <dom/net> <addr>   <MTU>
           # Random front end
           host   hyper.nsco.com   FF88   0103   3702   4148
           # because we want to show the 4 byte format
           host   192.12.102.1   FF00   0000   2203   1024
           # Small packets, interactive traffic.
           host   cray-b.nas.nasa.gov   FF88   0103   4401   4148
           # The other interface, for big packets.
           ahost   cray-b.nas.nasa.gov   FF88   0103   4501   32768
           # A loopback interface, (What else)
           loop   loop37.nsco.com   FF00   0000   3700   4148
           # And of course an example of arp service.
           arpserver hcgate.nsco.com   FF88   0103   7F07

    Comments may begin with either # or ;.
    Case is not significant in any field.

    <type> indicates the type of entity to be defined.
      Currently defined types are "host," "ahost", "loop," "address,"
      and "arpserver".

      host   This token indicates an IP host. The following field is
              expected to be a name that can be resolved to an IP
              address.

      ahost   This field indicates an additional network interface to
              the same host. This may be used for performance
              enhancements.

loop   Sets a flag in the entry for that host so that 0xFF00 is
         placed in bytes 8 and 9 of the message. This will cause
         the IP datagram to be directed towards the specified host
         (which must still be a valid host name) and looped back
         within the remote adapter. This facility serves both as a
         debugging aid and as a crude probe of the availability of
         the remote network adapter.

arpserver This indicates an address to use for directing ARP
         requests to the network. If several arpserver addresses
         are specified, they will be tried in turn until a response
         is received (or we run out of servers.) An arpserver with
         the appropriate broadcast address of FFFF FF07 would
         cause an ARP broadcast to take place when broadcasting
         becomes available. Broadcast and specific addresses may
         be used in combination.

<hostname> This field is the logical name of the destination. For a
host it is the logical name to be given to the local naming service
to determine the associated IP address. This field may contain four
decimal numbers separated by dots, in which case it is assumed to be
the explicit IP address.

<trunks/flags> This field is the value to be placed in bytes 0 and 1
of the message header when sending to this host. The associated data
bit need not be supplied as the driver must control it. All other
bits are sent as provided. This field is a hexidecimal number.

<domain/net> This field is the value to be placed in the Domain and
Network number field of the message. A value of 0000 in this field
indicates that the destination should be reached by constructing a
16-bit HYPERchannel header, rather than a 32-bit header.

<address> This field is the value to be placed in the 16-bit TO field
to reach <hostname>. This field is a hexidecimal number.

<MTU> This field contains the largest size IP datagram that the
destination host is prepared to receive. This field is a decimal
number. This field is optional. If not present, a value of 4148 is
assumed. See the earlier discussion on Maximum Transmission Unit for
more detail.

ARP SERVERS

The primary problem with local host address resolution is that
changes or additions to hosts on the local net must be replicated to
every HYPERchannel host in that network. While this is manageable
for up to half a dozen hosts, it becomes quite unmanageable for

larger networks. An approach that can be implemented using existing
HYPERchannel technology is to have a server on the HYPERchannel
network provide the HYPERchannel destination address that is
associated with an IP address.

Although this is strictly a point-to-point request/response dialogue
between two network nodes, the Address Resolution Protocol which was
originally designed for Ethernet (but thoughtfully constructed to
work with any pair of link and network addresses) performs an
excellent job.

ARP servers can be reached simply by placing the address of the
server in the 32-bit TO address of the network message. ARP servers
only "listen" to messages that arrive on their well known normal
address; they do not respond to ARP broadcast messages. Properly
equipped IP drivers should respond to the broadcast messages when
they appear.

If an ARP server receives a message containing an IP address it does
not know how to resolve, it ignores the message so that another ARP
server might be addressed at the source's next attempt.

If the address is resolvable, it places the known HYPERchannel
address and MTU size in the response and returns it to the location
in the HYPERchannel header FROM address.

Unlike a broadcast ARP, the ARP server will be required to service
two requests when two hosts that are initially unknown to one another
attempt to get in touch. Since the destination did not receive the
ARP request, it must contact the ARP server when its higher level
protocols first generate a datagram to respond to the the source's
first IP datagram to go through to the destination.

The source configuration file described in the previous section was
explicitly designed so that it could be sufficient as a data base for
an ARP server as well as an individual host.

BROADCAST ARP

When a local HYPERchannel network contains a broadcast capability,
any IP driver wishing to perform HYPERchannel address resolution may
be configured to emit the ARP message on a broadcast instead of a
well known address. IP drivers on other hosts are presumed to know
if their local HYPERchannel interface can send broadcast messages; if
so, they arrange to "listen" on the FF07 broadcast TO address for
ARP.

Processing of a received ARP broadcast message is otherwise identical

to RFC-826:

o   Messages are responded to if and only if the destination IP
     driver is authoritative for the designated IP address.

o   Whenever an ARP message is processed, the IP driver takes the
     source HYPERchannel address and MTU size and adds it to its
     address resolution tables. Thus the driver is equipped to
     turn around the IP datagram that arrives from the destination
     host when contact is made.

Each IP driver may have address resolutions that are set through a
static routing table (the configuration file specified above). If
ARP information arrives that contradicts a static entry (as opposed
to previously set dynamic ARP information) then the ARP information
should be ignored. This decision is made on the premise that the
only useful purpose of static routing in a broadcast ARP environment
is to add authentication, as it's easy to lie with ARP.

APPENDIX A. NSC PRODUCT ARCHITECTURE AND ADDRESSING

   This section is intended to be a concise review of the state of the
   art in NSC networks and the techniques they provide for the delivery
   of messages. Those who are thoroughly familiar with HYPERchannel may
   wish to only skim this section; however, there is material on new
   technologies and addressing formats that are not yet generally known
   to most of NSC's customers.

NETWORK SYSTEMS HYPERCHANNEL TECHNOLOGIES

   Network Systems manufactures several different network technologies
   that use very different media and link controls, but still provide a
   common host interface in both the protocol and hardware sense of the
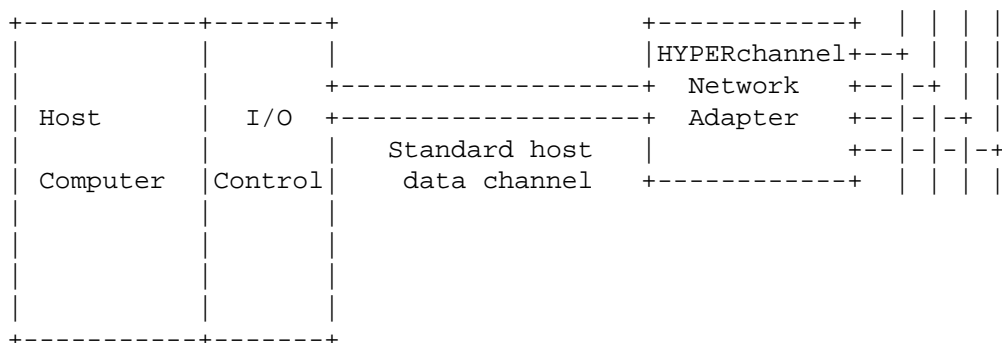   term. These four technologies are:

   o   HYPERchannel A -- A 50-megabit, baseband, CSMA with collision
       avoidance network using a coaxial cable bus. Individual
       HYPERchannel "network adapters" can control up to 4 of these
       coaxial cable "trunks," providing up to 200 megabits of
       capacity on a fully interconnected network. HYPERchannel A
       is NSC's earliest product and has been in production since
       1977. It is principally used to interconnect larger
       mainframe computers and high speed mainframe peripherals such
       as tape drives and laser printers.

   o   HYPERchannel B -- a 10-megabit, baseband, CSMA with collision
       avoidance network using a single coaxial cable bus. This
       technology is used for direct host to host communications under
       the name HYPERchannel B, and for terminal connections under the
       name HYPERbus. It is currently used for three major
       applications -- local networks of ASCII terminals, networks
       of IBM 3270 terminals, and host to host communications of
       smaller computers.

   o   DATAPIPE[3] -- a 275-megabit fiber optic "backbone" network
       that interconnects lower speed local area networks within a 20
       mile range, and to provide an ultra-high-performance network for
       the next generation of supercomputers and optical storage
       systems. A prototype version of DATApipe is currently under
       development at a customer site.

   o   Bridges and Network Distance Extensions -- NSC quickly
       discovered that its customers wanted very high speeds over
       geographic areas, not just within the range of several miles
       that is conceivable with a coaxial cable network. Starting
       in 1978, NSC began to build a series of "link adapters" that
       are integral bridges between local area networks. These link

adapters support common high speed communications media such
as Telco T1 circuits, private microwave, high speed
satellite links, and fiber optic point to point connections.

ATTACHMENT TO HOST COMPUTERS

   Network Systems' high speed interfaces use the attachment techniques
   of the manufacturer's highest speed peripheral controllers in order
   to achieve burst transfer rates of tens of megabits per second.
   These attachment techniques fall into three categories:

"MAINFRAME" DATA CHANNEL ATTACHMENT

```
     +----------+-------+                      +-----------+  | | | |
     |          |       |                      |HYPERchannel+--+ | | |
     |          |       +------------------+   Network    +--|-+ | |
     |  Host    |  I/O  +------------------+   Adapter    +--|-|-+ |
     |          |       |    Standard host |              +--|-|-|-+
     |  Computer|Control|     data channel  +-----------+  | | | |
     |          |       |                                  | | | |
     |          |       |                                  | | | |
     |          |       |                                  | | | |
     |          |       |                                  | | | |
     +----------+-------+
```

   The network adapter contains interface boards and firmware to be
   cabled to the manufacturer's data channel, such as would be done with
   a disk or tape controller. Mainframe network adapters do not emulate
   an existing manufacturer's device (such as a tape drive) but are
   supported by software which functions the channel and adapter to send
   and receive network messages.

   Models of HYPERchannel adapters are available for essentially all
   large scale computers worldwide.

MINICOMPUTER AND WORKSTATION ATTACHMENT

   Since the network adapter contains lots of expensive, high speed
   logic, a different technique is used to provide attachment to
   minicomputers and workstations.

```
    +-------------+       +--------------+       +--------------+
    |             |       |              |       |              |
    | Minicomputer|       |  Supermini   |       | Workstation  |
    |             |       |              |       |              |
    +-----+-------+       +-------+------+       +-------+------+
    |     | DMA   |       |       | DMA  |       | DMA   |      |
    |     |control|       |       |control|      |control|      |
    +-----+---++--+       +-------+--++---+      +--++---+------+
          ||                      ||                 ||
          ||                      ||                 ||
          |+---------+            ||      +--------+|
           +---------+|           ||      |+--------+
                     ||           ||      ||
          +-++--+-----+--++-+--++-+
          |    |     |    |    |
          +-----+-----+-----+-----+
          |         x400          |
          |    Network Adapter    |
          |                       |
          +-------+-+-+-+---------+
                  | | | |
     -----------------|-|-|-+---------------
     -----------------|-|-+-----------------
     -----------------|-+-------------------
     -----------------+---------------------
```
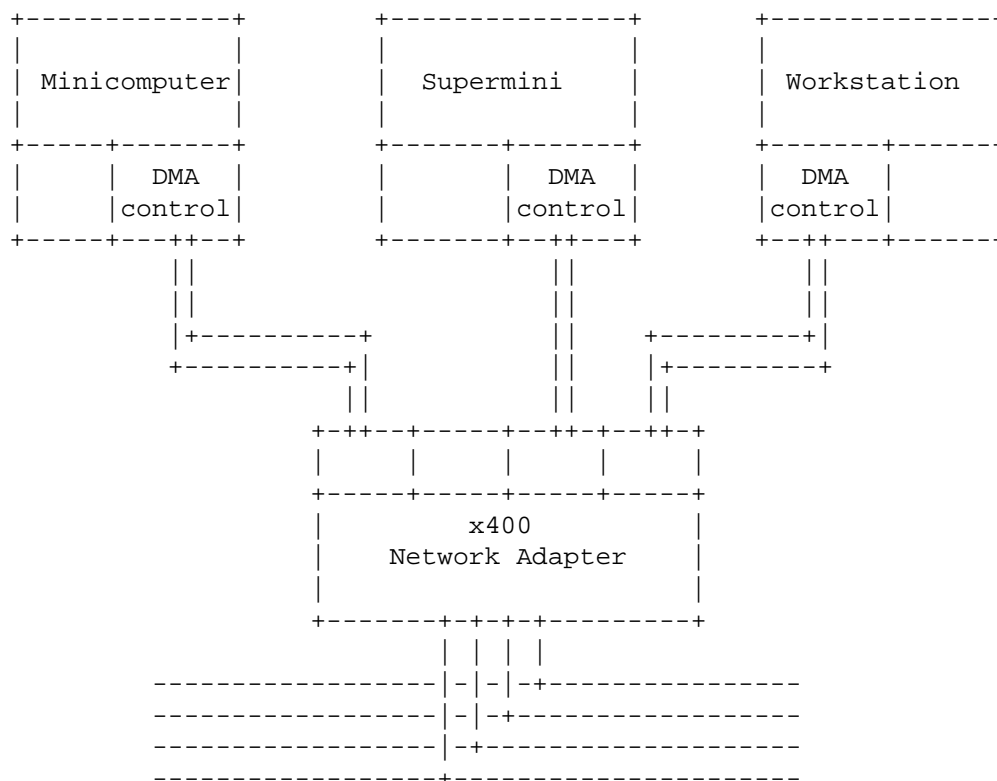
   In this case, NSC provides a DMA controller designed for direct
   connection to that minicomputer's backplane bus. These DMA
   controllers accept functions and burst blocks of data from host
   memory to a channel cable that is connected to one of four ports on a
   "general purpose computer adapter." This adapter multiplexes
   transmissions to and from the HYPERchannel trunks from up to four
   attached processors.

NETWORK COPROCESSORS

    For about 10 different bus systems, Network systems provides a
    "smart" DMA controller containing onboard memory and a Motorola 68010
    protocol processor.

```
        +------------+-----+---------------+-------+
        |            |     |  Coprocessor  |       |         +--------+
        |            |Host |    MC 68010   |Adapter+--------+  x400  |
        |    HOST    |DMA  |   256K memory |  DMA  +--------+ Adapter|
        |            |     |               |       |         +--------+
        |    Memory  +-----+---------------+-------+
        |            |
        +------------+
```
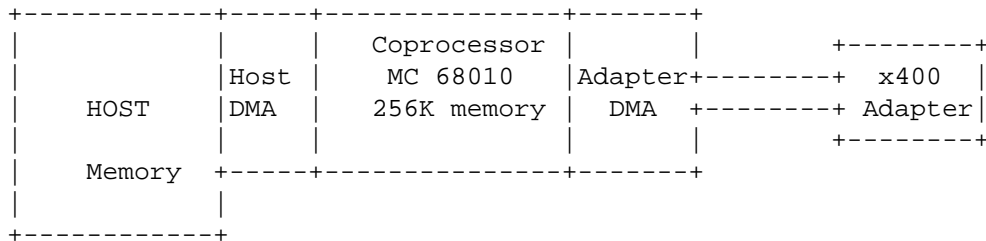
    This class of interface works through the network coprocessor's
    direct access to host memory. Network transmit and receive request
    packets are placed in a common "mailbox" area and extracted by the
    coprocessor. The coprocessor reads and writes system memory as
    required to service network requests in the proper order. The
    coprocessors currently provide a service to read or write network
    messages (called Driver service as it is more or less identical to
    HYPERchannel dumb DMA drivers) and a service for NETEX, which is
    NSC's OSI-like communications protocol.

APPENDIX B. NETWORK SYSTEMS HYPERCHANNEL PROTOCOLS

   The protocols implemented by NSC within its own boxes are designed
   for the needs of the different technologies. A compact summation of
   these protocols is:

```
     HYPERchannel B          HYPERchannel A           DATApipe
      10 Mbits/second       50-200 Mbits/second    275 Mbits/second
   +---------------------+---------------------+--------------------+
   |                                                               |
   |                  HYPERchannel network message                 |
   |                  connectionless datagram protocol             |
   |                                                               |
   +---------------------+---------------------+--------------------+
   |     "HYPERchannel   |                     |                    |
   |  compatibility mode"|   HYPERchannel A    |     DATApipe       |
   |    Virtual circuit  |   reservation and   |   acknowledgment   |
   |    estab. & control |    flow control     |    & flow control  |
   +---------------------+     protocol        |      protocol      |
   |                     |                     |                    |
   |  Virtual Circuits   |                     |                    |
   |    Flow Control     |                     |                    |
   +---------------------+---------------------+--------------------+
   |     CSMA / VT       |      CSMA / CA      |                    |
   |   frame (datagram)  |    frame (datagram) | TDMA packet delivery|
   |     delivery and    |     delivery and    |                    |
   |    acknowledgment   |    acknowledgment   |                    |
   +---------------------+---------------------+--------------------+
   |                     |                     |    Fiber optics    |
   |    75 ohm coax      |  1-4 75 ohm coax    | (various cable sizes|
   |       cable         |      cables         |  and xmission modes)|
   +---------------------+---------------------+--------------------+
```

   Without getting into great detail on these internal protocols, a few
   points are particularly interesting to system designers:

   o   All three technologies supply the same interface to the host
       computer or network coprocessor, a service to send and receive
       network messages that are datagrams from the host's vantage in
       that each contains sufficient information to deliver the message
       in and of itself. Since this datagram and its header fields are
       of paramount interest to the host implementor, it is discussed
       in detail below.

   o   All technologies use acknowledgments at a very low level to
       determine if packets have been successfully delivered. In
       addition to permitting a highly tuned contention mechanism for
       the coax medium, it also permits HYPERchannel A to balance the

load over several coax cables -- a feat that has proven very
difficult on, for example, Ethernet.

o   All boxes go to some lengths to assure that resources exist
    in the receiving box before actual transmission takes place.
    HYPERchannel B uses a virtual circuit that endures for several
    seconds of inactivity after one host first attempts to send a
    message to the other. Traffic over this "working virtual
    circuit" is flow controlled from source to destination and
    buffer resources are reserved for the path.

HYPERchannel A exchanges frames at very high rates to determine that
the receiver is ready to receive data and to control its flow as data
moves through the network.

DATApipe propagation time is relatively long compared to the time
needed to send an internal packet of 2K-4K bytes. As a result,
DATApipe controllers use a streamlined TP4-like transport protocol to
assure delivery of frames between DATApipe boxes.

REFERENCES

    [1]   HYPERchannel is a trademark of Network Systems Corporation.

    [2]   Plummer, D., "An Ethernet Address Resolution Protocol",
          RFC-826, Symbolics, September 1982.

    [3]   DATApipe is a registered trademark of Network Systems
          Corporation.