INTERNET PROTOCOL

DARPA INTERNET PROGRAM

PROTOCOL SPECIFICATION

September 1981

prepared for

Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

by

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, California 90291

TABLE OF CONTENTS

PREFACE iii

1. INTRODUCTION 1

1.1 Motivation 1
1.2 Scope 1
1.3 Interfaces 1
1.4 Operation 2

2. OVERVIEW 5

2.1 Relation to Other Protocols 9
2.2 Model of Operation 5
2.3 Function Description 7
2.4 Gateways 9

3. SPECIFICATION 11

3.1 Internet Header Format 11
3.2 Discussion 23
3.3 Interfaces 31

APPENDIX A: Examples & Scenarios 34
APPENDIX B: Data Transmission Order 39

GLOSSARY 41

REFERENCES 45

PREFACE

This document specifies the DoD Standard Internet Protocol. This document is based on six earlier editions of the ARPA Internet Protocol Specification, and the present text draws heavily from them. There have been many contributors to this work both in terms of concepts and in terms of text. This edition revises aspects of addressing, error handling, option codes, and the security, precedence, compartments, and handling restriction features of the internet protocol.

Jon Postel

Editor

RFC: 791
Replaces: RFC 760
IENs 128, 123, 111,
80, 54, 44, 41, 28, 26

INTERNET PROTOCOL

DARPA INTERNET PROGRAM
PROTOCOL SPECIFICATION

1. INTRODUCTION

1.1. Motivation

The Internet Protocol is designed for use in interconnected systems of packet-switched computer communication networks. Such a system has been called a "catenet" [1]. The internet protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses. The internet protocol also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through "small packet" networks.

1.2. Scope

The internet protocol is specifically limited in scope to provide the functions necessary to deliver a package of bits (an internet datagram) from a source to a destination over an interconnected system of networks. There are no mechanisms to augment end-to-end data reliability, flow control, sequencing, or other services commonly found in host-to-host protocols. The internet protocol can capitalize on the services of its supporting networks to provide various types and qualities of service.

1.3. Interfaces

This protocol is called on by host-to-host protocols in an internet environment. This protocol calls on local network protocols to carry the internet datagram to the next gateway or destination host.

For example, a TCP module would call on the internet module to take a TCP segment (including the TCP header and user data) as the data portion of an internet datagram. The TCP module would provide the addresses and other parameters in the internet header to the internet module as arguments of the call. The internet module would then create an internet datagram and call on the local network interface to transmit the internet datagram.

In the ARPANET case, for example, the internet module would call on a local net module which would add the 1822 leader [2] to the internet datagram creating an ARPANET message to transmit to the IMP. The ARPANET address would be derived from the internet address by the local network interface and would be the address of some host in the ARPANET, that host might be a gateway to other networks.

1.4. Operation

The internet protocol implements two basic functions: addressing and fragmentation.

The internet modules use the addresses carried in the internet header to transmit internet datagrams toward their destinations. The selection of a path for transmission is called routing.

The internet modules use fields in the internet header to fragment and reassemble internet datagrams when necessary for transmission through "small packet" networks.

The model of operation is that an internet module resides in each host engaged in internet communication and in each gateway that interconnects networks. These modules share common rules for interpreting address fields and for fragmenting and assembling internet datagrams. In addition, these modules (especially in gateways) have procedures for making routing decisions and other functions.

The internet protocol treats each internet datagram as an independent entity unrelated to any other internet datagram. There are no connections or logical circuits (virtual or otherwise).

The internet protocol uses four key mechanisms in providing its service: Type of Service, Time to Live, Options, and Header Checksum.

The Type of Service is used to indicate the quality of the service desired. The type of service is an abstract or generalized set of parameters which characterize the service choices provided in the networks that make up the internet. This type of service indication is to be used by gateways to select the actual transmission parameters for a particular network, the network to be used for the next hop, or the next gateway when routing an internet datagram.

The Time to Live is an indication of an upper bound on the lifetime of an internet datagram. It is set by the sender of the datagram and reduced at the points along the route where it is processed. If the time to live reaches zero before the internet datagram reaches its destination, the internet datagram is destroyed. The time to live can be thought of as a self destruct time limit.

The Options provide for control functions needed or useful in some situations but unnecessary for the most common communications. The options include provisions for timestamps, security, and special routing.

The Header Checksum provides a verification that the information used in processing internet datagram has been transmitted correctly. The data may contain errors. If the header checksum fails, the internet datagram is discarded at once by the entity which detects the error.

The internet protocol does not provide a reliable communication facility. There are no acknowledgments either end-to-end or hop-by-hop. There is no error control for data, only a header checksum. There are no retransmissions. There is no flow control.

Errors detected may be reported via the Internet Control Message Protocol (ICMP) [3] which is implemented in the internet protocol module.

2. OVERVIEW

2.1. Relation to Other Protocols

The following diagram illustrates the place of the internet protocol in the protocol hierarchy:

```
 +------+ +-----+ +-----+     +-----+
 |Telnet| | FTP | | TFTP| ... | ... |
 +------+ +-----+ +-----+     +-----+
       |   |         |           |
      +-----+     +-----+     +-----+
      | TCP |     | UDP | ... | ... |
      +-----+     +-----+     +-----+
         |           |           |
      +--------------------------+----+
      |    Internet Protocol & ICMP   |
      +--------------------------+----+
                     |
        +---------------------------+
        |   Local Network Protocol  |
        +---------------------------+
```

Protocol Relationships

Figure 1.

Internet protocol interfaces on one side to the higher level host-to-host protocols and on the other side to the local network protocol. In this context a "local network" may be a small network in a building or a large network such as the ARPANET.

2.2. Model of Operation

The model of operation for transmitting a datagram from one application program to another is illustrated by the following scenario:

We suppose that this transmission will involve one intermediate gateway.

The sending application program prepares its data and calls on its local internet module to send that data as a datagram and passes the destination address and other parameters as arguments of the call.

The internet module prepares a datagram header and attaches the data to it. The internet module determines a local network address for this internet address, in this case it is the address of a gateway.

It sends this datagram and the local network address to the local network interface.

The local network interface creates a local network header, and attaches the datagram to it, then sends the result via the local network.

The datagram arrives at a gateway host wrapped in the local network header, the local network interface strips off this header, and turns the datagram over to the internet module. The internet module determines from the internet address that the datagram is to be forwarded to another host in a second network. The internet module determines a local net address for the destination host. It calls on the local network interface for that network to send the datagram.

This local network interface creates a local network header and attaches the datagram sending the result to the destination host.

At this destination host the datagram is stripped of the local net header by the local network interface and handed to the internet module.

The internet module determines that the datagram is for an application program in this host. It passes the data to the application program in response to a system call, passing the source address and other parameters as results of the call.

```
Application                                           Application
Program                                                   Program
      \                                                   /
    Internet Module      Internet Module      Internet Module
          \                 /       \                /
          LNI-1          LNI-1      LNI-2         LNI-2
             \           /             \          /
            Local Network 1           Local Network 2
```

Transmission Path

Figure 2

2.3. Function Description

The function or purpose of Internet Protocol is to move datagrams through an interconnected set of networks. This is done by passing the datagrams from one internet module to another until the destination is reached. The internet modules reside in hosts and gateways in the internet system. The datagrams are routed from one internet module to another through individual networks based on the interpretation of an internet address. Thus, one important mechanism of the internet protocol is the internet address.

In the routing of messages from one internet module to another, datagrams may need to traverse a network whose maximum packet size is smaller than the size of the datagram. To overcome this difficulty, a fragmentation mechanism is provided in the internet protocol.

Addressing

A distinction is made between names, addresses, and routes [4]. A name indicates what we seek. An address indicates where it is. A route indicates how to get there. The internet protocol deals primarily with addresses. It is the task of higher level (i.e., host-to-host or application) protocols to make the mapping from names to addresses. The internet module maps internet addresses to local net addresses. It is the task of lower level (i.e., local net or gateways) procedures to make the mapping from local net addresses to routes.

Addresses are fixed length of four octets (32 bits). An address begins with a network number, followed by local address (called the "rest" field). There are three formats or classes of internet addresses: in class a, the high order bit is zero, the next 7 bits are the network, and the last 24 bits are the local address; in class b, the high order two bits are one-zero, the next 14 bits are the network and the last 16 bits are the local address; in class c, the high order three bits are one-one-zero, the next 21 bits are the network and the last 8 bits are the local address.

Care must be taken in mapping internet addresses to local net addresses; a single physical host must be able to act as if it were several distinct hosts to the extent of using several distinct internet addresses. Some hosts will also have several physical interfaces (multi-homing).

That is, provision must be made for a host to have several physical interfaces to the network with each having several logical internet addresses.

Examples of address mappings may be found in "Address Mappings" [5].

Fragmentation

Fragmentation of an internet datagram is necessary when it originates in a local net that allows a large packet size and must traverse a local net that limits packets to a smaller size to reach its destination.

An internet datagram can be marked "don't fragment." Any internet datagram so marked is not to be internet fragmented under any circumstances. If internet datagram marked don't fragment cannot be delivered to its destination without fragmenting it, it is to be discarded instead.

Fragmentation, transmission and reassembly across a local network which is invisible to the internet protocol module is called intranet fragmentation and may be used [6].

The internet fragmentation and reassembly procedure needs to be able to break a datagram into an almost arbitrary number of pieces that can be later reassembled. The receiver of the fragments uses the identification field to ensure that fragments of different datagrams are not mixed. The fragment offset field tells the receiver the position of a fragment in the original datagram. The fragment offset and length determine the portion of the original datagram covered by this fragment. The more-fragments flag indicates (by being reset) the last fragment. These fields provide sufficient information to reassemble datagrams.

The identification field is used to distinguish the fragments of one datagram from those of another. The originating protocol module of an internet datagram sets the identification field to a value that must be unique for that source-destination pair and protocol for the time the datagram will be active in the internet system. The originating protocol module of a complete datagram sets the more-fragments flag to zero and the fragment offset to zero.

To fragment a long internet datagram, an internet protocol module (for example, in a gateway), creates two new internet datagrams and copies the contents of the internet header fields from the long datagram into both new internet headers. The data of the long datagram is divided into two portions on a 8 octet (64 bit) boundary (the second portion might not be an integral multiple of 8 octets, but the first must be). Call the number of 8 octet blocks in the first portion NFB (for Number of Fragment Blocks). The first portion of the data is placed in the first new internet datagram, and the total length field is set to the length of the first datagram. The more-fragments flag is set to one. The second portion of the data is placed in the second new internet datagram, and the total length field is set to the length of the second datagram. The more-fragments flag carries the same value as the long datagram. The fragment offset field of the second new internet datagram is set to the value of that field in the long datagram plus NFB.

This procedure can be generalized for an n-way split, rather than the two-way split described.

To assemble the fragments of an internet datagram, an internet protocol module (for example at a destination host) combines internet datagrams that all have the same value for the four fields: identification, source, destination, and protocol. The combination is done by placing the data portion of each fragment in the relative position indicated by the fragment offset in that fragment's internet header. The first fragment will have the fragment offset zero, and the last fragment will have the more-fragments flag reset to zero.

2.4. Gateways

Gateways implement internet protocol to forward datagrams between networks. Gateways also implement the Gateway to Gateway Protocol (GGP) [7] to coordinate routing and other internet control information.

In a gateway the higher level protocols need not be implemented and the GGP functions are added to the IP module.

```
 +-------------------------------+
 | Internet Protocol & ICMP & GGP|
 +-------------------------------+
         |                 |
+---------------+   +---------------+
|   Local Net   |   |   Local Net   |
+---------------+   +---------------+
```

Gateway Protocols

Figure 3.

3. SPECIFICATION

3.1. Internet Header Format

A summary of the contents of the internet header follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Example Internet Datagram Header

Figure 4.

Note that each tick mark represents one bit position.

Version: 4 bits

The Version field indicates the format of the internet header. This document describes version 4.

IHL: 4 bits

Internet Header Length is the length of the internet header in 32 bit words, and thus points to the beginning of the data. Note that the minimum value for a correct header is 5.

Type of Service: 8 bits

The Type of Service provides an indication of the abstract parameters of the quality of service desired. These parameters are to be used to guide the selection of the actual service parameters when transmitting a datagram through a particular network. Several networks offer service precedence, which somehow treats high precedence traffic as more important than other traffic (generally by accepting only traffic above a certain precedence at time of high load). The major choice is a three way tradeoff between low-delay, high-reliability, and high-throughput.

```
Bits 0-2:  Precedence.
Bit    3:  0 = Normal Delay,      1 = Low Delay.
Bits   4:  0 = Normal Throughput, 1 = High Throughput.
Bits   5:  0 = Normal Relibility, 1 = High Relibility.
Bit  6-7:  Reserved for Future Use.

   0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+-----+
|                 |     |     |     |     |     |
|   PRECEDENCE    |  D  |  T  |  R  |  0  |  0  |
|                 |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+

  Precedence

    111 - Network Control
    110 - Internetwork Control
    101 - CRITIC/ECP
    100 - Flash Override
    011 - Flash
    010 - Immediate
    001 - Priority
    000 - Routine
```

The use of the Delay, Throughput, and Reliability indications may increase the cost (in some sense) of the service. In many networks better performance for one of these parameters is coupled with worse performance on another. Except for very unusual cases at most two of these three indications should be set.

The type of service is used to specify the treatment of the datagram during its transmission through the internet system. Example mappings of the internet type of service to the actual service provided on networks such as AUTODIN II, ARPANET, SATNET, and PRNET is given in "Service Mappings" [8].

The Network Control precedence designation is intended to be used within a network only. The actual use and control of that designation is up to each network. The Internetwork Control designation is intended for use by gateway control originators only. If the actual use of these precedence designations is of concern to a particular network, it is the responsibility of that network to control the access to, and use of, those precedence designations.

Total Length: 16 bits

Total Length is the length of the datagram, measured in octets, including internet header and data. This field allows the length of a datagram to be up to 65,535 octets. Such long datagrams are impractical for most hosts and networks. All hosts must be prepared to accept datagrams of up to 576 octets (whether they arrive whole or in fragments). It is recommended that hosts only send datagrams larger than 576 octets if they have assurance that the destination is prepared to accept the larger datagrams.

The number 576 is selected to allow a reasonable sized data block to be transmitted in addition to the required header information. For example, this size allows a data block of 512 octets plus 64 header octets to fit in a datagram. The maximal internet header is 60 octets, and a typical internet header is 20 octets, allowing a margin for headers of higher level protocols.

Identification: 16 bits

An identifying value assigned by the sender to aid in assembling the fragments of a datagram.

Flags: 3 bits

Various Control Flags.

```
  Bit 0: reserved, must be zero
  Bit 1: (DF) 0 = May Fragment,  1 = Don't Fragment.
  Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.

      0   1   2
    +---+---+---+
    |   | D | M |
    | 0 | F | F |
    +---+---+---+
```

Fragment Offset: 13 bits

This field indicates where in the datagram this fragment belongs.

The fragment offset is measured in units of 8 octets (64 bits). The first fragment has offset zero.

Time to Live: 8 bits

This field indicates the maximum time the datagram is allowed to remain in the internet system. If this field contains the value zero, then the datagram must be destroyed. This field is modified in internet header processing. The time is measured in units of seconds, but since every module that processes a datagram must decrease the TTL by at least one even if it process the datagram in less than a second, the TTL must be thought of only as an upper bound on the time a datagram may exist. The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum datagram lifetime.

Protocol: 8 bits

This field indicates the next level protocol used in the data portion of the internet datagram. The values for various protocols are specified in "Assigned Numbers" [9].

Header Checksum: 16 bits

A checksum on the header only. Since some header fields change (e.g., time to live), this is recomputed and verified at each point that the internet header is processed.

The checksum algorithm is:

> The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

This is a simple to compute checksum and experimental evidence indicates it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Source Address: 32 bits

The source address. See section 3.2.

Destination Address: 32 bits

The destination address. See section 3.2.

Options: variable

The options may appear or not in datagrams. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular datagram, not their implementation.

In some environments the security option may be required in all datagrams.

The option field is variable in length. There may be zero or more options. There are two cases for the format of an option:

```
Case 1:  A single octet of option-type.

Case 2:  An option-type octet, an option-length octet, and the
         actual option-data octets.
```

The option-length octet counts the option-type octet and the option-length octet as well as the option-data octets.

The option-type octet is viewed as having 3 fields:

```
  1 bit   copied flag,
  2 bits  option class,
  5 bits  option number.

The copied flag indicates that this option is copied into all
fragments on fragmentation.

  0 = not copied
  1 = copied

The option classes are:

  0 = control
  1 = reserved for future use
  2 = debugging and measurement
  3 = reserved for future use


The following internet options are defined:

  CLASS NUMBER LENGTH DESCRIPTION
  ----- ------ ------ -----------
    0     0      -    End of Option list.  This option occupies only
                      1 octet; it has no length octet.
    0     1      -    No Operation.  This option occupies only 1
                      octet; it has no length octet.
    0     2     11    Security.  Used to carry Security,
                      Compartmentation, User Group (TCC), and
                      Handling Restriction Codes compatible with DOD
                      requirements.
    0     3     var.  Loose Source Routing.  Used to route the
                      internet datagram based on information
                      supplied by the source.
    0     9     var.  Strict Source Routing.  Used to route the
                      internet datagram based on information
                      supplied by the source.
    0     7     var.  Record Route.  Used to trace the route an
                      internet datagram takes.
    0     8      4    Stream ID.  Used to carry the stream
                      identifier.
    2     4     var.  Internet Timestamp.



Specific Option Definitions

  End of Option List

    +--------+
    |00000000|
    +--------+
      Type=0

    This option indicates the end of the option list.  This might
    not coincide with the end of the internet header according to
    the internet header length.  This is used at the end of all
```

options, not the end of each option, and need only be used if the end of the options would not otherwise coincide with the end of the internet header.

May be copied, introduced, or deleted on fragmentation, or for any other reason.

No Operation

```
+--------+
|00000001|
+--------+
  Type=1
```

This option may be used between options, for example, to align the beginning of a subsequent option on a 32 bit boundary.

May be copied, introduced, or deleted on fragmentation, or for any other reason.

Security

This option provides a way for hosts to send security, compartmentation, handling restrictions, and TCC (closed user group) parameters. The format for this option is as follows:

```
+--------+--------+---//---+---//---+---//---+---//---+
|10000010|00001011|SSS  SSS|CCC  CCC|HHH  HHH|  TCC   |
+--------+--------+---//---+---//---+---//---+---//---+
 Type=130 Length=11
```

Security (S field): 16 bits

Specifies one of 16 levels of security (eight of which are reserved for future use).

```
00000000 00000000 - Unclassified
11110001 00110101 - Confidential
01111000 10011010 - EFTO
10111100 01001101 - MMMM
01011110 00100110 - PROG
10101111 00010011 - Restricted
11010111 10001000 - Secret
01101011 11000101 - Top Secret
00110101 11100010 - (Reserved for future use)
10011010 11110001 - (Reserved for future use)
01001101 01111000 - (Reserved for future use)
00100100 10111101 - (Reserved for future use)
00010011 01011110 - (Reserved for future use)
10001001 10101111 - (Reserved for future use)
11000100 11010110 - (Reserved for future use)
11100010 01101011 - (Reserved for future use)
```

Compartments (C field): 16 bits

An all zero value is used when the information transmitted is not compartmented. Other values for the compartments field may be obtained from the Defense Intelligence Agency.

Handling Restrictions (H field): 16 bits

The values for the control and release markings are alphanumeric digraphs and are defined in the Defense Intelligence Agency Manual DIAM 65-19, "Standard Security Markings".

Transmission Control Code (TCC field): 24 bits

Provides a means to segregate traffic and define controlled communities of interest among subscribers. The TCC values are trigraphs, and are available from HQ DCA Code 530.

Must be copied on fragmentation. This option appears at most once in a datagram.

Loose Source and Record Route

```
+--------+--------+--------+---------//--------+
|10000011| length | pointer|     route data    |
+--------+--------+--------+---------//--------+
 Type=131
```

The loose source and record route (LSRR) option provides a means for the source of an internet datagram to supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information.

The option begins with the option type code. The second octet is the option length which includes the option type code and the length octet, the pointer octet, and length-3 octets of route data. The third octet is the pointer into the route data indicating the octet which begins the next source address to be processed. The pointer is relative to this option, and the smallest legal value for the pointer is 4.

A route data is composed of a series of internet addresses. Each internet address is 32 bits or 4 octets. If the pointer is greater than the length, the source route is empty (and the recorded route full) and the routing is to be based on the destination address field.

If the address in destination address field has been reached and the pointer is not greater than the length, the next address in the source route replaces the address in the destination address field, and the recorded route address replaces the source address just used, and pointer is increased by four.

The recorded route address is the internet module's own internet address as known in the environment into which this datagram is being forwarded.

This procedure of replacing the source route with the recorded route (though it is in the reverse of the order it must be in to be used as a source route) means the option (and the IP header as a whole) remains a constant length as the datagram progresses through the internet.

This option is a loose source route because the gateway or host IP is allowed to use any route of any number of other intermediate gateways to reach the next address in the route.

Must be copied on fragmentation. Appears at most once in a datagram.

Strict Source and Record Route

```
+--------+--------+--------+---------//--------+
|10001001| length | pointer|     route data    |
+--------+--------+--------+---------//--------+
 Type=137
```

The strict source and record route (SSRR) option provides a means for the source of an internet datagram to supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information.

The option begins with the option type code. The second octet is the option length which includes the option type code and the length octet, the pointer octet, and length-3 octets of route data. The third octet is the pointer into the route data indicating the octet which begins the next source address to be processed. The pointer is relative to this option, and the smallest legal value for the pointer is 4.

A route data is composed of a series of internet addresses. Each internet address is 32 bits or 4 octets. If the pointer is greater than the length, the source route is empty (and the recorded route full) and the routing is to be based on the destination address field.

If the address in destination address field has been reached and the pointer is not greater than the length, the next address in the source route replaces the address in the destination address field, and the recorded route address replaces the source address just used, and pointer is increased by four.

The recorded route address is the internet module's own internet address as known in the environment into which this datagram is being forwarded.

This procedure of replacing the source route with the recorded route (though it is in the reverse of the order it must be in to be used as a source route) means the option (and the IP header as a whole) remains a constant length as the datagram progresses through the internet.

This option is a strict source route because the gateway or host IP must send the datagram directly to the next address in the source route through only the directly connected network indicated in the next address to reach the next gateway or host specified in the route.

Must be copied on fragmentation. Appears at most once in a datagram.

Record Route

```
+--------+--------+--------+---------//--------+
|00000111| length | pointer|     route data    |
+--------+--------+--------+---------//--------+
  Type=7
```

The record route option provides a means to record the route of an internet datagram.

The option begins with the option type code. The second octet is the option length which includes the option type code and the length octet, the pointer octet, and length-3 octets of route data. The third octet is the pointer into the route data indicating the octet which begins the next area to store a route address. The pointer is relative to this option, and the smallest legal value for the pointer is 4.

A recorded route is composed of a series of internet addresses. Each internet address is 32 bits or 4 octets. If the pointer is greater than the length, the recorded route data area is full. The originating host must compose this option with a large enough route data area to hold all the address expected. The size of the option does not change due to adding addresses. The intitial contents of the route data area must be zero.

When an internet module routes a datagram it checks to see if the record route option is present. If it is, it inserts its own internet address as known in the environment into which this datagram is being forwarded into the recorded route begining at the octet indicated by the pointer, and increments the pointer by four.

If the route data area is already full (the pointer exceeds the length) the datagram is forwarded without inserting the address into the recorded route. If there is some room but not enough room for a full address to be inserted, the original datagram is considered to be in error and is discarded. In either case an ICMP parameter problem message may be sent to the source host [3].

Not copied on fragmentation, goes in first fragment only. Appears at most once in a datagram.

Stream Identifier

```
+--------+--------+--------+--------+
|10001000|00000010|    Stream ID    |
+--------+--------+--------+--------+
 Type=136 Length=4
```

This option provides a way for the 16-bit SATNET stream identifier to be carried through networks that do not support the stream concept.

Must be copied on fragmentation. Appears at most once in a datagram.

Internet Timestamp

```
+--------+--------+--------+--------+
|01000100| length | pointer|oflw|flg|
+--------+--------+--------+--------+
|         internet address          |
+--------+--------+--------+--------+
|             timestamp             |
+--------+--------+--------+--------+
|                 .                 |
                  .
                  .
Type = 68
```

The Option Length is the number of octets in the option counting the type, length, pointer, and overflow/flag octets (maximum length 40).

The Pointer is the number of octets from the beginning of this option to the end of timestamps plus one (i.e., it points to the octet beginning the space for next timestamp). The smallest legal value is 5. The timestamp area is full when the pointer is greater than the length.

The Overflow (oflw) [4 bits] is the number of IP modules that cannot register timestamps due to lack of space.

The Flag (flg) [4 bits] values are

0 -- time stamps only, stored in consecutive 32-bit words,

1 -- each timestamp is preceded with internet address of the registering entity,

3 -- the internet address fields are prespecified. An IP module only registers its timestamp if it matches its own address with the next specified internet address.

The Timestamp is a right-justified, 32-bit timestamp in milliseconds since midnight UT. If the time is not available in milliseconds or cannot be provided with respect to midnight UT then any time may be inserted as a timestamp provided the high order bit of the timestamp field is set to one to indicate the use of a non-standard value.

The originating host must compose this option with a large enough timestamp data area to hold all the timestamp information expected. The size of the option does not change due to adding timestamps. The intitial contents of the timestamp data area must be zero or internet address/zero pairs.
If the timestamp data area is already full (the pointer exceeds the length) the datagram is forwarded without inserting the timestamp, but the overflow count is incremented by one.

If there is some room but not enough room for a full timestamp to be inserted, or the overflow count itself overflows, the original datagram is considered to be in error and is discarded. In either case an ICMP parameter problem message may be sent to the source host [3].

The timestamp option is not copied upon fragmentation. It is carried in the first fragment. Appears at most once in a datagram.

Padding: variable

The internet header padding is used to ensure that the internet header ends on a 32 bit boundary. The padding is zero.

3.2. Discussion

The implementation of a protocol must be robust. Each implementation must expect to interoperate with others created by different individuals. While the goal of this specification is to be explicit about the protocol there is the possibility of differing interpretations. In general, an implementation must be conservative in its sending behavior, and liberal in its receiving behavior. That is, it must be careful to send well-formed datagrams, but must accept any datagram that it can interpret (e.g., not object to technical errors where the meaning is still clear).

The basic internet service is datagram oriented and provides for the fragmentation of datagrams at gateways, with reassembly taking place at the destination internet protocol module in the destination host. Of course, fragmentation and reassembly of datagrams within a network or by private agreement between the gateways of a network is also allowed since this is transparent to the internet protocols and the higher-level protocols. This transparent type of fragmentation and reassembly is termed "network-dependent" (or intranet) fragmentation and is not discussed further here.

Internet addresses distinguish sources and destinations to the host level and provide a protocol field as well. It is assumed that each protocol will provide for whatever multiplexing is necessary within a host.

Addressing

To provide for flexibility in assigning address to networks and allow for the large number of small to intermediate sized networks the interpretation of the address field is coded to specify a small number of networks with a large number of host, a moderate number of networks with a moderate number of hosts, and a large number of networks with a small number of hosts. In addition there is an escape code for extended addressing mode.

Address Formats:

High Order Bits	Format	Class
0	7 bits of net, 24 bits of host	a
10	14 bits of net, 16 bits of host	b
110	21 bits of net, 8 bits of host	c
111	escape to extended addressing mode	

A value of zero in the network field means this network. This is only used in certain ICMP messages. The extended addressing mode is undefined. Both of these features are reserved for future use.

The actual values assigned for network addresses is given in "Assigned Numbers" [9].

The local address, assigned by the local network, must allow for a single physical host to act as several distinct internet hosts. That is, there must be a mapping between internet host addresses and network/host interfaces that allows several internet addresses to correspond to one interface. It must also be allowed for a host to have several physical interfaces and to treat the datagrams from several of them as if they were all addressed to a single host.

Address mappings between internet addresses and addresses for ARPANET, SATNET, PRNET, and other networks are described in "Address Mappings" [5].

Fragmentation and Reassembly.

The internet identification field (ID) is used together with the source and destination address, and the protocol fields, to identify datagram fragments for reassembly.

The More Fragments flag bit (MF) is set if the datagram is not the last fragment. The Fragment Offset field identifies the fragment location, relative to the beginning of the original unfragmented datagram. Fragments are counted in units of 8 octets. The fragmentation strategy is designed so than an unfragmented datagram has all zero fragmentation information (MF = 0, fragment offset = 0). If an internet datagram is fragmented, its data portion must be broken on 8 octet boundaries.

This format allows 2**13 = 8192 fragments of 8 octets each for a total of 65,536 octets. Note that this is consistent with the the datagram total length field (of course, the header is counted in the total length and not in the fragments).

When fragmentation occurs, some options are copied, but others remain with the first fragment only.

Every internet module must be able to forward a datagram of 68 octets without further fragmentation. This is because an internet header may be up to 60 octets, and the minimum fragment is 8 octets.

Every internet destination must be able to receive a datagram of 576 octets either in one piece or in fragments to be reassembled.

The fields which may be affected by fragmentation include:

(1) options field
(2) more fragments flag
(3) fragment offset
(4) internet header length field
(5) total length field
(6) header checksum

If the Don't Fragment flag (DF) bit is set, then internet fragmentation of this datagram is NOT permitted, although it may be discarded. This can be used to prohibit fragmentation in cases where the receiving host does not have sufficient resources to reassemble internet fragments.

One example of use of the Don't Fragment feature is to down line load a small host. A small host could have a boot strap program that accepts a datagram stores it in memory and then executes it.

The fragmentation and reassembly procedures are most easily described by examples. The following procedures are example implementations.

General notation in the following pseudo programs: "=<" means "less than or equal", "#" means "not equal", "=" means "equal", "<-" means "is set to". Also, "x to y" includes x and excludes y; for example, "4 to 7" would include 4, 5, and 6 (but not 7).

An Example Fragmentation Procedure

The maximum sized datagram that can be transmitted through the next network is called the maximum transmission unit (MTU).

If the total length is less than or equal the maximum transmission unit then submit this datagram to the next step in datagram processing; otherwise cut the datagram into two fragments, the first fragment being the maximum size, and the second fragment being the rest of the datagram. The first fragment is submitted to the next step in datagram processing, while the second fragment is submitted to this procedure in case it is still too large.

Notation:

```
FO    -  Fragment Offset
IHL   -  Internet Header Length
DF    -  Don't Fragment flag
MF    -  More Fragments flag
TL    -  Total Length
OFO   -  Old Fragment Offset
OIHL  -  Old Internet Header Length
```

```
  OMF   -  Old More Fragments flag
  OTL   -  Old Total Length
  NFB   -  Number of Fragment Blocks
  MTU   -  Maximum Transmission Unit
```

Procedure:

```
  IF TL =< MTU THEN Submit this datagram to the next step
       in datagram processing ELSE IF DF = 1 THEN discard the
  datagram ELSE
  To produce the first fragment:
  (1)  Copy the original internet header;
  (2)  OIHL <- IHL; OTL <- TL; OFO <- FO; OMF <- MF;
  (3)  NFB <- (MTU-IHL*4)/8;
  (4)  Attach the first NFB*8 data octets;
  (5)  Correct the header:
       MF <- 1;  TL <- (IHL*4)+(NFB*8);
       Recompute Checksum;
  (6)  Submit this fragment to the next step in
       datagram processing;
  To produce the second fragment:
  (7)  Selectively copy the internet header (some options
       are not copied, see option definitions);
  (8)  Append the remaining data;
  (9)  Correct the header:
       IHL <- (((OIHL*4)-(length of options not copied))+3)/4;
       TL <- OTL - NFB*8 - (OIHL-IHL)*4);
       FO <- OFO + NFB;  MF <- OMF;  Recompute Checksum;
  (10) Submit this fragment to the fragmentation test; DONE.
```

In the above procedure each fragment (except the last) was made the maximum allowable size. An alternative might produce less than the maximum size datagrams. For example, one could implement a fragmentation procedure that repeatly divided large datagrams in half until the resulting fragments were less than the maximum transmission unit size.

An Example Reassembly Procedure

For each datagram the buffer identifier is computed as the concatenation of the source, destination, protocol, and identification fields. If this is a whole datagram (that is both the fragment offset and the more fragments fields are zero), then any reassembly resources associated with this buffer identifier are released and the datagram is forwarded to the next step in datagram processing.

If no other fragment with this buffer identifier is on hand then reassembly resources are allocated. The reassembly resources consist of a data buffer, a header buffer, a fragment block bit table, a total data length field, and a timer. The data from the fragment is placed in the data buffer according to its fragment offset and length, and bits are set in the fragment block bit table corresponding to the fragment blocks received.

If this is the first fragment (that is the fragment offset is zero) this header is placed in the header buffer. If this is the last fragment (that is the more fragments field is zero) the total data length is computed. If this fragment completes the datagram (tested by checking the bits set in the fragment block table), then the datagram is sent to the next step in datagram processing; otherwise the timer is set to the maximum of the current timer value and the value of the time to live field from this fragment; and the reassembly routine gives up control.

If the timer runs out, the all reassembly resources for this buffer identifier are released. The initial setting of the timer is a lower bound on the reassembly waiting time. This is because the waiting time will be increased if the Time to Live in the arriving fragment is greater than the current timer value but will not be decreased if it is less. The maximum this timer value could reach is the maximum time to live (approximately 4.25 minutes). The current recommendation for the initial timer setting is 15 seconds. This may be changed as experience with this protocol accumulates. Note that the choice of this parameter value is related to the buffer capacity available and the data rate of the transmission medium; that is, data rate times timer value equals buffer size (e.g., 10Kb/s X 15s = 150Kb).

Notation:

```
  FO    -  Fragment Offset
  IHL   -  Internet Header Length
  MF    -  More Fragments flag
  TTL   -  Time To Live
  NFB   -  Number of Fragment Blocks
  TL    -  Total Length
  TDL   -  Total Data Length
  BUFID -  Buffer Identifier
  RCVBT -  Fragment Received Bit Table
  TLB   -  Timer Lower Bound
```

Procedure:

```
  (1)  BUFID <- source|destination|protocol|identification;
  (2)  IF FO = 0 AND MF = 0
  (3)     THEN IF buffer with BUFID is allocated
  (4)             THEN flush all reassembly for this BUFID;
  (5)          Submit datagram to next step; DONE.
  (6)     ELSE IF no buffer with BUFID is allocated
  (7)             THEN allocate reassembly resources
                       with BUFID;
                       TIMER <- TLB; TDL <- 0;
  (8)          put data from fragment into data buffer with
               BUFID from octet FO*8 to
                                   octet (TL-(IHL*4))+FO*8;
  (9)          set RCVBT bits from FO
                                  to FO+((TL-(IHL*4)+7)/8);
  (10)         IF MF = 0 THEN TDL <- TL-(IHL*4)+(FO*8)
  (11)         IF FO = 0 THEN put header in header buffer
  (12)         IF TDL # 0
  (13)          AND all RCVBT bits from 0
                                       to (TDL+7)/8 are set
  (14)            THEN TL <- TDL+(IHL*4)
```

```
(15)                 Submit datagram to next step;
(16)                 free all reassembly resources
                     for this BUFID; DONE.
(17)         TIMER <- MAX(TIMER,TTL);
(18)         give up until next fragment or timer expires;
(19) timer expires: flush all reassembly with this BUFID; DONE.
```

In the case that two or more fragments contain the same data either identically or through a partial overlap, this procedure will use the more recently arrived copy in the data buffer and datagram delivered.

Identification

The choice of the Identifier for a datagram is based on the need to provide a way to uniquely identify the fragments of a particular datagram. The protocol module assembling fragments judges fragments to belong to the same datagram if they have the same source, destination, protocol, and Identifier. Thus, the sender must choose the Identifier to be unique for this source, destination pair and protocol for the time the datagram (or any fragment of it) could be alive in the internet.

It seems then that a sending protocol module needs to keep a table of Identifiers, one entry for each destination it has communicated with in the last maximum packet lifetime for the internet.

However, since the Identifier field allows 65,536 different values, some host may be able to simply use unique identifiers independent of destination.

It is appropriate for some higher level protocols to choose the identifier. For example, TCP protocol modules may retransmit an identical TCP segment, and the probability for correct reception would be enhanced if the retransmission carried the same identifier as the original transmission since fragments of either datagram could be used to construct a correct TCP segment.

Type of Service

The type of service (TOS) is for internet service quality selection. The type of service is specified along the abstract parameters precedence, delay, throughput, and reliability. These abstract parameters are to be mapped into the actual service parameters of the particular networks the datagram traverses.

Precedence. An independent measure of the importance of this datagram.

Delay. Prompt delivery is important for datagrams with this indication.

Throughput. High data rate is important for datagrams with this indication.

Reliability. A higher level of effort to ensure delivery is important for datagrams with this indication.

For example, the ARPANET has a priority bit, and a choice between "standard" messages (type 0) and "uncontrolled" messages (type 3), (the choice between single packet and multipacket messages can also be considered a service parameter). The uncontrolled messages tend to be less reliably delivered and suffer less delay. Suppose an internet datagram is to be sent through the ARPANET. Let the internet type of service be given as:

```
Precedence:    5
Delay:         0
Throughput:    1
Reliability:   1
```

In this example, the mapping of these parameters to those available for the ARPANET would be to set the ARPANET priority bit on since the Internet precedence is in the upper half of its range, to select standard messages since the throughput and reliability requirements are indicated and delay is not. More details are given on service mappings in "Service Mappings" [8].

Time to Live

The time to live is set by the sender to the maximum time the datagram is allowed to be in the internet system. If the datagram is in the internet system longer than the time to live, then the datagram must be destroyed.

This field must be decreased at each point that the internet header is processed to reflect the time spent processing the datagram. Even if no local information is available on the time actually spent, the field must be decremented by 1. The time is measured in units of seconds (i.e. the value 1 means one second). Thus, the maximum time to live is 255 seconds or 4.25 minutes. Since every module that processes a datagram must decrease the TTL by at least one even if it process the datagram in less than a second, the TTL must be thought of only as an upper bound on the time a datagram may exist. The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum datagram lifetime.

Some higher level reliable connection protocols are based on assumptions that old duplicate datagrams will not arrive after a certain time elapses. The TTL is a way for such protocols to have an assurance that their assumption is met.

Options

The options are optional in each datagram, but required in implementations. That is, the presence or absence of an option is the choice of the sender, but each internet module must be able to parse every option. There can be several options present in the option field.

The options might not end on a 32-bit boundary. The internet header must be filled out with octets of zeros. The first of these would be interpreted as the end-of-options option, and the remainder as internet header padding.

Every internet module must be able to act on every option. The Security Option is required if classified, restricted, or compartmented traffic is to be passed.

Checksum

The internet header checksum is recomputed if the internet header is changed. For example, a reduction of the time to live, additions or changes to internet options, or due to fragmentation. This checksum at the internet level is intended to protect the internet header fields from transmission errors.

There are some applications where a few data bit errors are acceptable while retransmission delays are not. If the internet protocol enforced data correctness such applications could not be supported.

Errors

Internet protocol errors may be reported via the ICMP messages [3].

3.3. Interfaces

The functional description of user interfaces to the IP is, at best, fictional, since every operating system will have different facilities. Consequently, we must warn readers that different IP implementations may have different user interfaces. However, all IPs must provide a certain minimum set of services to guarantee that all IP implementations can support the same protocol hierarchy. This section specifies the functional interfaces required of all IP implementations.

Internet protocol interfaces on one side to the local network and on the other side to either a higher level protocol or an application program. In the following, the higher level protocol or application program (or even a gateway program) will be called the "user" since it is using the internet module. Since internet protocol is a datagram protocol, there is minimal memory or state maintained between datagram transmissions, and each call on the internet protocol module by the user supplies all information necessary for the IP to perform the service requested.

An Example Upper Level Interface

The following two example calls satisfy the requirements for the user to internet protocol module communication ("=>" means returns):

SEND (src, dst, prot, TOS, TTL, BufPTR, len, Id, DF, opt => result)

where:

src = source address
dst = destination address
prot = protocol

TOS = type of service
TTL = time to live
BufPTR = buffer pointer
len = length of buffer
Id = Identifier
DF = Don't Fragment
opt = option data
result = response
OK = datagram sent ok
Error = error in arguments or local network error

Note that the precedence is included in the TOS and the security/compartment is passed as an option.

RECV (BufPTR, prot, => result, src, dst, TOS, len, opt)

where:

BufPTR = buffer pointer
prot = protocol
result = response
OK = datagram received ok
Error = error in arguments
len = length of buffer
src = source address
dst = destination address
TOS = type of service
opt = option data

When the user sends a datagram, it executes the SEND call supplying all the arguments. The internet protocol module, on receiving this call, checks the arguments and prepares and sends the message. If the arguments are good and the datagram is accepted by the local network, the call returns successfully. If either the arguments are bad, or the datagram is not accepted by the local network, the call returns unsuccessfully. On unsuccessful returns, a reasonable report must be made as to the cause of the problem, but the details of such reports are up to individual implementations.

When a datagram arrives at the internet protocol module from the local network, either there is a pending RECV call from the user addressed or there is not. In the first case, the pending call is satisfied by passing the information from the datagram to the user. In the second case, the user addressed is notified of a pending datagram. If the user addressed does not exist, an ICMP error message is returned to the sender, and the data is discarded.

The notification of a user may be via a pseudo interrupt or similar mechanism, as appropriate in the particular operating system environment of the implementation.

A user's RECV call may then either be immediately satisfied by a pending datagram, or the call may be pending until a datagram arrives.

The source address is included in the send call in case the sending host has several addresses (multiple physical connections or logical addresses). The internet module must check to see that the source address is one of the legal address for this host.

An implementation may also allow or require a call to the internet module to indicate interest in or reserve exclusive use of a class of datagrams (e.g., all those with a certain value in the protocol field).

This section functionally characterizes a USER/IP interface. The notation used is similar to most procedure of function calls in high level languages, but this usage is not meant to rule out trap type service calls (e.g., SVCs, UUOs, EMTs), or any other form of interprocess communication.

APPENDIX A: Examples & Scenarios

Example 1:

This is an example of the minimal data carrying internet datagram:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver= 4 |IHL= 5 |Type of Service|        Total Length = 21      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Identification = 111     |Flg=0|   Fragment Offset = 0   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Time = 123  |  Protocol = 1 |        header checksum        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         source address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      destination address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     data      |
+-+-+-+-+-+-+-+-+
```

Example Internet Datagram

Figure 5.

Note that each tick mark represents one bit position.

This is a internet datagram in version 4 of internet protocol; the internet header consists of five 32 bit words, and the total length of the datagram is 21 octets. This datagram is a complete datagram (not a fragment).

Example 2:

In this example, we show first a moderate size internet datagram (452 data octets), then two internet fragments that might result from the fragmentation of this datagram if the maximum sized transmission allowed were 280 octets.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver= 4 |IHL= 5 |Type of Service|       Total Length = 472      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Identification = 111      |Flg=0|     Fragment Offset = 0 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Time = 123  | Protocol = 6  |        header checksum        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         source address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      destination address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
\                                                               \
\                                                               \
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             data              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Example Internet Datagram

Figure 6.

Now the first fragment that results from splitting the datagram after 256 data octets.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver= 4 |IHL= 5 |Type of Service|       Total Length = 276      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Identification = 111      |Flg=1|     Fragment Offset = 0 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Time = 119  | Protocol = 6  |        Header Checksum        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         source address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      destination address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
\                                                               \
\                                                               \
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Example Internet Fragment

Figure 7.

And the second fragment.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver= 4 |IHL= 5 |Type of Service|       Total Length = 216      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Identification = 111      |Flg=0|  Fragment Offset  =  32 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Time = 119  | Protocol = 6  |        Header Checksum        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         source address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      destination address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
\                                                               \
\                                                               \
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            data               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Example Internet Fragment

Figure 8.

Example 3:

Here, we show an example of a datagram containing options:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver= 4 |IHL= 8 |Type of Service|       Total Length = 576      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Identification = 111    |Flg=0|     Fragment Offset = 0 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Time = 123  |  Protocol = 6 |       Header Checksum         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        source address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      destination address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Opt. Code = x | Opt.  Len.= 3 | option value  | Opt. Code = x |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Opt. Len. = 4 |           option value        | Opt. Code = 1 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Opt. Code = y | Opt. Len. = 3 |  option value | Opt. Code = 0 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
\                                                               \
\                                                               \
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Example Internet Datagram

Figure 9.

APPENDIX B: Data Transmission Order

The order of transmission of the header and data described in this document is resolved to the octet level. Whenever a diagram shows a group of octets, the order of transmission of those octets is the normal order in which they are read in English. For example, in the following diagram the octets are transmitted in the order they are numbered.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       1       |       2       |       3       |       4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       5       |       6       |       7       |       8       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       9       |      10       |      11       |      12       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Transmission Order of Bytes

Figure 10.

Whenever an octet represents a numeric quantity the left most bit in the diagram is the high order or most significant bit. That is, the bit labeled 0 is the most significant bit. For example, the following diagram represents the value 170 (decimal).

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|1 0 1 0 1 0 1 0|
+-+-+-+-+-+-+-+-+
```

Significance of Bits

Figure 11.

Similarly, whenever a multi-octet field represents a numeric quantity the left most bit of the whole field is the most significant bit. When a multi-octet quantity is transmitted the most significant octet is transmitted first.

GLOSSARY

1822
: BBN Report 1822, "The Specification of the Interconnection of a Host and an IMP". The specification of interface between a host and the ARPANET.

ARPANET leader
: The control information on an ARPANET message at the host-IMP interface.

ARPANET message
: The unit of transmission between a host and an IMP in the ARPANET. The maximum size is about 1012 octets (8096 bits).

ARPANET packet
: A unit of transmission used internally in the ARPANET between IMPs. The maximum size is about 126 octets (1008 bits).

Destination
: The destination address, an internet header field.

DF
: The Don't Fragment bit carried in the flags field.

Flags
: An internet header field carrying various control flags.

Fragment Offset
: This internet header field indicates where in the internet datagram a fragment belongs.

GGP
Gateway to Gateway Protocol, the protocol used primarily between gateways to control routing and other gateway functions.

header
Control information at the beginning of a message, segment, datagram, packet or block of data.

ICMP
Internet Control Message Protocol, implemented in the internet module, the ICMP is used from gateways to hosts and between hosts to report errors and make routing suggestions.

Identification
An internet header field carrying the identifying value assigned by the sender to aid in assembling the fragments of a datagram.

IHL
The internet header field Internet Header Length is the length of the internet header measured in 32 bit words.

IMP
The Interface Message Processor, the packet switch of the ARPANET.

Internet Address
A four octet (32 bit) source or destination address consisting of a Network field and a Local Address field.

internet datagram
The unit of data exchanged between a pair of internet modules (includes the internet header).

internet fragment
A portion of the data of an internet datagram with an internet header.

Local Address
The address of a host within a network. The actual mapping of an internet local address on to the host addresses in a network is quite general, allowing for many to one mappings.

MF
The More-Fragments Flag carried in the internet header flags field.

module
An implementation, usually in software, of a protocol or other procedure.

more-fragments flag
A flag indicating whether or not this internet datagram contains the end of an internet datagram, carried in the internet header Flags field.

NFB

The Number of Fragment Blocks in a the data portion of an internet fragment. That is, the length of a portion of data measured in 8 octet units.

octet

An eight bit byte.

Options

The internet header Options field may contain several options, and each option may be several octets in length.

Padding

The internet header Padding field is used to ensure that the data begins on 32 bit word boundary. The padding is zero.

Protocol

In this document, the next higher level protocol identifier, an internet header field.

Rest

The local address portion of an Internet Address.

Source

The source address, an internet header field.

TCP

Transmission Control Protocol: A host-to-host protocol for reliable communication in internet environments.

TCP Segment

The unit of data exchanged between TCP modules (including the TCP header).

TFTP

Trivial File Transfer Protocol: A simple file transfer protocol built on UDP.

Time to Live

An internet header field which indicates the upper bound on how long this internet datagram may exist.

TOS

Type of Service

Total Length

The internet header field Total Length is the length of the datagram in octets including internet header and data.

TTL

Time to Live

Type of Service

An internet header field which indicates the type (or quality) of service for this internet datagram.

UDP

User Datagram Protocol: A user level protocol for transaction oriented applications.

User

The user of the internet protocol. This may be a higher level protocol module, an application program, or a gateway program.

Version

The Version field indicates the format of the internet header.

REFERENCES

[1] Cerf, V., "The Catenet Model for Internetworking," Information Processing Techniques Office, Defense Advanced Research Projects Agency, IEN 48, July 1978.

[2] Bolt Beranek and Newman, "Specification for the Interconnection of a Host and an IMP," BBN Technical Report 1822, Revised May 1978.

[3] Postel, J., "Internet Control Message Protocol - DARPA Internet Program Protocol Specification," RFC 792, USC/Information Sciences Institute, September 1981.

[4] Shoch, J., "Inter-Network Naming, Addressing, and Routing," COMPCON, IEEE Computer Society, Fall 1978.

[5] Postel, J., "Address Mappings," RFC 796, USC/Information Sciences Institute, September 1981.

[6] Shoch, J., "Packet Fragmentation in Inter-Network Protocols," Computer Networks, v. 3, n. 1, February 1979.

[7] Strazisar, V., "How to Build a Gateway", IEN 109, Bolt Beranek and Newman, August 1979.

[8] Postel, J., "Service Mappings," RFC 795, USC/Information Sciences Institute, September 1981.

[9] Postel, J., "Assigned Numbers," RFC 790, USC/Information Sciences Institute, September 1981.

==

Network Working Group   J. Mogul (Stanford)
Request for Comments: 950   J. Postel (ISI)
                                                                    August 1985

Internet Standard Subnetting Procedure

Status Of This Memo

This RFC specifies a protocol for the ARPA-Internet community. If subnetting is implemented it is strongly recommended that these procedures be followed. Distribution of this memo is unlimited.

Overview

This memo discusses the utility of "subnets" of Internet networks, which are logically visible sub-sections of a single Internet network. For administrative or technical reasons, many organizations have chosen to divide one Internet network into several subnets, instead of acquiring a set of Internet network numbers. This memo specifies procedures for the use of subnets. These procedures are for hosts (e.g., workstations). The procedures used in and between subnet gateways are not fully described. Important motivation and background information for a subnetting standard is provided in RFC-940 [7].

Acknowledgment

This memo is based on RFC-917 [1]. Many people contributed to the development of the concepts described here. J. Noel Chiappa, Chris Kent, and Tim Mann, in particular, provided important suggestions. Additional contributions in shaping this memo were made by Zaw-Sing Su, Mike Karels, and the Gateway Algorithms and Data Structures Task Force (GADS).

1. Motivation

The original view of the Internet universe was a two-level hierarchy: the top level the Internet as a whole, and the level below it individual networks, each with its own network number. The Internet does not have a hierarchical topology, rather the interpretation of addresses is hierarchical. In this two-level model, each host sees its network as a single entity; that is, the network may be treated as a "black box" to which a set of hosts is connected.

While this view has proved simple and powerful, a number of organizations have found it inadequate, and have added a third level to the interpretation of Internet addresses. In this view, a given Internet network is divided into a collection of subnets.

The three-level model is useful in networks belonging to moderately large organizations (e.g., Universities or companies with more than one building), where it is often necessary to use more than one LAN cable to cover a "local area". Each LAN may then be treated as a subnet.

There are several reasons why an organization might use more than one cable to cover a campus:

- Different technologies: Especially in a research environment, there may be more than one kind of LAN in use; e.g., an organization may have some equipment that supports Ethernet, and some that supports a ring network.

- Limits of technologies: Most LAN technologies impose limits, based on electrical parameters, on the number of hosts connected, and on the total length of the cable. It is easy to exceed these limits, especially those on cable length.

- Network congestion: It is possible for a small subset of the hosts on a LAN to monopolize most of the bandwidth. A common solution to this problem is to divide the hosts into cliques of high mutual communication, and put these cliques on separate cables.

- Point-to-Point links: Sometimes a "local area", such as a university campus, is split into two locations too far apart to connect using the preferred LAN technology. In this case, high-speed point-to-point links might connect several LANs.

An organization that has been forced to use more than one LAN has three choices for assigning Internet addresses:

1. Acquire a distinct Internet network number for each cable; subnets are not used at all.

2. Use a single network number for the entire organization, but assign host numbers without regard to which LAN a host is on ("transparent subnets").

3. Use a single network number, and partition the host address space by assigning subnet numbers to the LANs ("explicit subnets").

Each of these approaches has disadvantages. The first, although not requiring any new or modified protocols, results in an explosion in the size of Internet routing tables. Information about the internal details of local connectivity is propagated everywhere, although it is of little or no use outside the local organization. Especially as some current gateway implementations do not have much space for routing tables, it would be good to avoid this problem.

The second approach requires some convention or protocol that makes the collection of LANs appear to be a single Internet network. For example, this can be done on LANs where each Internet address is translated to a hardware address using an Address Resolution Protocol (ARP), by having the bridges between the LANs intercept ARP requests for non-local targets, see RFC-925 [2]. However, it is not possible to do this for all LAN technologies, especially those where ARP protocols are not currently used, or if the LAN does not support broadcasts. A more fundamental problem is that bridges must discover which LAN a host is on, perhaps by using a broadcast algorithm. As the number of LANs grows, the cost of broadcasting grows as well; also, the size of translation caches required in the bridges grows with the total number of hosts in the network.

The third approach is to explicitly support subnets. This does have a disadvantage, in that it is a modification of the Internet Protocol, and thus requires changes to IP implementations already in use (if these implementations are to be used on a subnetted network). However, these changes are relatively minor, and once made, yield a simple and efficient solution to the problem. Also, the approach avoids any changes that would be incompatible with existing hosts on non-subnetted networks.

Further, when appropriate design choices are made, it is possible for hosts which believe they are on a non-subnetted network to be used on a subnetted one, as explained in RFC-917 [1]. This is useful when it is not possible to modify some of the hosts to support subnets explicitly, or when a gradual transition is preferred.

2. Standards for Subnet Addressing

This section first describes a proposal for interpretation of Internet addresses to support subnets. Next it discusses changes to host software to support subnets. Finally, it presents a procedures for discovering what address interpretation is in use on a given network (i.e., what address mask is in use).

2.1. Interpretation of Internet Addresses

Suppose that an organization has been assigned an Internet network number, has further divided that network into a set of subnets, and wants to assign host addresses: how should this be done? Since there are minimal restrictions on the assignment of the "local address" part of the Internet address, several approaches have been proposed for representing the subnet number:

1. Variable-width field: Any number of the bits of the local address part are used for the subnet number; the size of this field, although constant for a given network, varies from network to network. If the field width is zero, then subnets are not in use.

2. Fixed-width field: A specific number of bits (e.g., eight) is used for the subnet number, if subnets are in use.

3. Self-encoding variable-width field: Just as the width (i.e., class) of the network number field is encoded by its high-order bits, the width of the subnet field is similarly encoded.

4. Self-encoding fixed-width field: A specific number of bits is used for the subnet number.

5. Masked bits: Use a bit mask ("address mask") to identify

which bits of the local address field indicate the subnet number.

What criteria can be used to choose one of these five schemes? First, should we use a self-encoding scheme? And, should it be possible to tell from examining an Internet address if it refers to a subnetted network, without reference to any other information?

An interesting feature of self-encoding is that it allows the address space of a network to be divided into subnets of different sizes, typically one subnet of half the address space and a set of small subnets.

For example, consider a class C network that uses a self-encoding scheme with one bit to indicate if it is the large subnet or not and an additional three bits to identify the small subnet. If the first bit is zero then this is the large subnet, if the first bit is one then the following bits (3 in this example) give the subnet number. There is one subnet with 128 host addresses, and eight subnets with 16 hosts each.

To establish a subnetting standard the parameters and interpretation of the self-encoding scheme must be fixed and consistent throughout the Internet.

It could be assumed that all networks are subnetted. This would allow addresses to be interpreted without reference to any other information.

This is a significant advantage, that given the Internet address no additional information is needed for an implementation to determine if two addresses are on the same subnet. However, this can also be viewed as a disadvantage: it may cause problems for networks which have existing host numbers that use arbitrary bits in the local address part. In other words, it is useful to be able to control whether a network is subnetted independently from the assignment of host addresses.

The alternative is to have the fact that a network is subnetted kept separate from the address. If one finds, somehow, that the network is subnetted then the standard self-encoded subnetted network address rules are followed, otherwise the non-subnetted network addressing rules are followed.

If a self-encoding scheme is not used, there is no reason to use a fixed-width field scheme: since there must in any case be some per-network "flag" to indicate if subnets are in use, the additional cost of using an integer (a subnet field width or address mask) instead of a boolean is negligible. The advantage of using the address mask scheme is that it allows each organization to choose the best way to allocate relatively scarce bits of local address to subnet and host numbers. Therefore, we choose the address-mask scheme: it is the most flexible scheme, yet costs no more to implement than any other.

For example, the Internet address might be interpreted as:

<network-number><subnet-number><host-number>

where the <network-number> field is as defined by IP [3], the <host-number> field is at least 1-bit wide, and the width of the <subnet-number> field is constant for a given network. No further structure is required for the <subnet-number> or <host-number> fields. If the width of the <subnet-number> field is zero, then the network is not subnetted (i.e., the interpretation of [3] is used).

For example, on a Class B network with a 6-bit wide subnet field, an address would be broken down like this:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0|        NETWORK            |  SUBNET   |    Host Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Since the bits that identify the subnet are specified by a bitmask, they need not be adjacent in the address. However, we recommend that the subnet bits be contiguous and located as the most significant bits of the local address.

Special Addresses:

From the Assigned Numbers memo [9]:

> "In certain contexts, it is useful to have fixed addresses with functional significance rather than as identifiers of specific hosts. When such usage is called for, the address zero is to be interpreted as meaning "this", as in "this network". The address of all ones are to be interpreted as meaning "all", as in "all hosts". For example, the address 128.9.255.255 could be interpreted as meaning all hosts on the network 128.9. Or, the address 0.0.0.37 could be interpreted as meaning host 37 on this network."

It is useful to preserve and extend the interpretation of these special addresses in subnetted networks. This means the values of all zeros and all ones in the subnet field should not be assigned to actual (physical) subnets.

In the example above, the 6-bit wide subnet field may have any value except 0 and 63.

Please note that there is no effect or new restriction on the addresses of hosts on non-subnetted networks.

2.2. Changes to Host Software to Support Subnets

In most implementations of IP, there is code in the module that handles outgoing datagrams to decide if a datagram can be sent directly to the destination on the local network or if it must be sent to a gateway.

Generally the code is something like this:

```
IF ip_net_number(dg.ip_dest) = ip_net_number(my_ip_addr)
    THEN
        send_dg_locally(dg, dg.ip_dest)
    ELSE
        send_dg_locally(dg,
                          gateway_to(ip_net_number(dg.ip_dest)))
```

(If the code supports multiply-connected networks, it will be more complicated, but this is irrelevant to the current discussion.)

To support subnets, it is necessary to store one more 32-bit quantity, called my_ip_mask. This is a bit-mask with bits set in the fields corresponding to the IP network number, and additional bits set corresponding to the subnet number field.

The code then becomes:

```
IF bitwise_and(dg.ip_dest, my_ip_mask)
                          = bitwise_and(my_ip_addr, my_ip_mask)
    THEN
        send_dg_locally(dg, dg.ip_dest)
    ELSE
        send_dg_locally(dg,
               gateway_to(bitwise_and(dg.ip_dest, my_ip_mask)))
```

Of course, part of the expression in the conditional can be pre-computed.

It may or may not be necessary to modify the "gateway_to" function, so that it too takes the subnet field bits into account when performing comparisons.

To support multiply-connected hosts, the code can be changed to keep the "my_ip_addr" and "my_ip_mask" quantities on a per-interface basis; the expression in the conditional must then be evaluated for each interface.

2.3. Finding the Address Mask

How can a host determine what address mask is in use on a subnet to which it is connected? The problem is analogous to several other "bootstrapping" problems for Internet hosts: how a host determines its own address, and how it locates a gateway on its local network. In all three cases, there are two basic solutions: "hardwired" information, and broadcast-based protocols.

Hardwired information is that available to a host in isolation from a network. It may be compiled-in, or (preferably) stored in a disk file. However, for the increasingly common case of a diskless workstation that is bootloaded over a LAN, neither hardwired solution is satisfactory.

Instead, since most LAN technology supports broadcasting, a better method is for the newly-booted host to broadcast a request for the necessary information. For example, for the purpose of determining its Internet address, a host may use the "Reverse Address Resolution Protocol" (RARP) [4].

However, since a newly-booted host usually needs to gather several facts (e.g., its IP address, the hardware address of a gateway, the IP address of a domain name server, the subnet address mask), it would be better to acquire all this information in one request if possible, rather than doing numerous broadcasts on the network. The mechanisms designed to boot diskless workstations can also load per-host specific configuration files that contain the required information (e.g., see RFC-951 [8]). It is possible, and desirable, to obtain all the facts necessary to operate a host from a boot server using only one broadcast message.

In the case where it is necessary for a host to find the address mask as a separate operation the following mechanism is provided:

> To provide the address mask information the ICMP protocol [5] is extended by adding a new pair of ICMP message types, "Address Mask Request" and "Address Mask Reply", analogous to the "Information Request" and "Information Reply" ICMP messages. These are described in detail in Appendix I.
>
> The intended use of these new ICMP messages is that a host, when booting, broadcast an "Address Mask Request" message. A gateway (or a host acting in lieu of a gateway) that receives this message responds with an "Address Mask Reply". If there is no indication in the request which host sent it (i.e., the IP Source Address is zero), the reply is broadcast as well. The requesting host will hear the response, and from it determine the address mask.
>
> Since there is only one possible value that can be sent in an "Address Mask Reply" on any given LAN, there is no need for the requesting host to match the responses it hears against the request it sent; similarly, there is no problem if more than one gateway responds. We assume that hosts reboot infrequently, so the broadcast load on a network from use of this protocol should be small.

If a host is connected to more than one LAN, it might have to find the address mask for each.

One potential problem is what a host should do if it can not find out the address mask, even after a reasonable number of tries. Three interpretations can be placed on the situation:

1. The local net exists in (permanent) isolation from all other nets.

2. Subnets are not in use, and no host can supply the address mask.

3. All gateways on the local net are (temporarily) down.

The first and second situations imply that the address mask is identical with the Internet network number mask. In the third situation, there is no way to determine what the proper value is; the safest choice is thus a mask identical with the Internet network number mask. Although this might later turn out to be wrong, it will not prevent transmissions that would otherwise succeed. It is possible for a host to recover from a wrong choice: when a gateway comes up, it should broadcast an "Address Mask Reply"; when a host receives such a message that disagrees with its guess, it should change its mask to conform to the received value. No host or gateway should send an "Address Mask Reply" based on a "guessed" value.

Finally, note that no host is required to use this ICMP protocol to discover the address mask; it is perfectly reasonable for a host with non-volatile storage to use stored information (including a configuration file from a boot server).

Appendix I. Address Mask ICMP

Address Mask Request or Address Mask Reply

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |      Code     |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Identifier          |       Sequence Number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Address Mask                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Fields:

Addresses

The address of the source in an address mask request message will be the destination of the address mask reply message. To form an address mask reply message, the source address of the request becomes the destination address of the reply, the source address of the reply is set to the replier's address, the type code changed to AM2, the address mask value inserted into the Address Mask field, and the checksum recomputed. However, if the source address in the request message is zero, then the destination address for the reply message should denote a broadcast.

ICMP Fields:

Type

AM1 for address mask request message

AM2 for address mask reply message

Code

0 for address mask request message

0 for address mask reply message

Checksum

The checksum is the 16-bit one's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum, the checksum field should be zero. This checksum may be replaced in the future.

Identifier

An identifier to aid in matching requests and replies, may be zero.

Sequence Number

A sequence number to aid in matching requests and replies, may be zero.

Address Mask

A 32-bit mask.

Description

A gateway receiving an address mask request should return it with the address mask field set to the 32-bit mask of the bits identifying the subnet and network, for the subnet on which the request was received.

If the requesting host does not know its own IP address, it may leave the source field zero; the reply should then be broadcast. However, this approach should be avoided if at all possible, since it increases the superfluous broadcast load on the network. Even when the replies are broadcast, since there is only one possible address mask for a subnet, there is no need to match requests with replies. The "Identifier" and "Sequence Number" fields can be ignored.

Type AM1 may be received from a gateway or a host.

Type AM2 may be received from a gateway, or a host acting in lieu of a gateway.

Appendix II. Examples

These examples show how a host can find out the address mask using the ICMP Address Mask Request and Address Mask Reply messages. For the following examples, assume that address 255.255.255.255 denotes "broadcast to this physical medium" [6].

1. A Class A Network Case

For this case, assume that the requesting host is on class A network 36.0.0.0, has address 36.40.0.123, that there is a gateway at 36.40.0.62, and that a 8-bit wide subnet field is in use, that is, the address mask is 255.255.0.0.

The most efficient method, and the one we recommend, is for a host to first discover its own address (perhaps using "RARP" [4]), and then to send the ICMP request to 255.255.255.255:

```
Source address:          36.40.0.123
Destination address:     255.255.255.255
Protocol:                ICMP = 1
Type:                    Address Mask Request = AM1
Code:                    0
Mask:                    0
```

The gateway can then respond directly to the requesting host.

```
Source address:          36.40.0.62
Destination address:     36.40.0.123
Protocol:                ICMP = 1
Type:                    Address Mask Reply = AM2
Code:                    0
Mask:                    255.255.0.0
```

Suppose that 36.40.0.123 is a diskless workstation, and does not know even its own host number. It could send the following datagram:

```
Source address:          0.0.0.0
Destination address:     255.255.255.255
Protocol:                ICMP = 1
Type:                    Address Mask Request = AM1
Code:                    0
Mask:                    0
```

36.40.0.62 will hear the datagram, and should respond with this datagram:

```
Source address:          36.40.0.62
Destination address:     255.255.255.255
Protocol:                ICMP = 1
Type:                    Address Mask Reply = AM2
Code:                    0
Mask:                    255.255.0.0
```

Note that the gateway uses the narrowest possible broadcast to reply. Even so, the over use of broadcasts presents an unnecessary load to all hosts on the subnet, and so the use of the "anonymous" (0.0.0.0) source address must be kept to a minimum.

If broadcasting is not allowed, we assume that hosts have wired-in information about neighbor gateways; thus, 36.40.0.123 might send this datagram:

```
Source address:          36.40.0.123
Destination address:     36.40.0.62
Protocol:                ICMP = 1
```

```
   Type:                     Address Mask Request = AM1
   Code:                     0
   Mask:                     0
```

36.40.0.62 should respond exactly as in the previous case.

```
   Source address:           36.40.0.62
   Destination address:      36.40.0.123
   Protocol:                 ICMP = 1
   Type:                     Address Mask Reply = AM2
   Code:                     0
   Mask:                     255.255.0.0
```

2. A Class B Network Case

For this case, assume that the requesting host is on class B network 128.99.0.0, has address 128.99.4.123, that there is a gateway at 128.99.4.62, and that a 6-bit wide subnet field is in use, that is, the address mask is 255.255.252.0.

The host sends the ICMP request to 255.255.255.255:

```
   Source address:           128.99.4.123
   Destination address:      255.255.255.255
   Protocol:                 ICMP = 1
   Type:                     Address Mask Request = AM1
   Code:                     0
   Mask:                     0
```

The gateway can then respond directly to the requesting host.

```
   Source address:           128.99.4.62
   Destination address:      128.99.4.123
   Protocol:                 ICMP = 1
   Type:                     Address Mask Reply = AM2
   Code:                     0
   Mask:                     255.255.252.0
```

In the diskless workstation case the host sends:

```
   Source address:           0.0.0.0
   Destination address:      255.255.255.255
   Protocol:                 ICMP = 1
   Type:                     Address Mask Request = AM1
   Code:                     0
   Mask:                     0
```

128.99.4.62 will hear the datagram, and should respond with this datagram:

```
   Source address:           128.99.4.62
   Destination address:      255.255.255.255
   Protocol:                 ICMP = 1
   Type:                     Address Mask Reply = AM2
   Code:                     0
   Mask:                     255.255.252.0
```

If broadcasting is not allowed 128.99.4.123 sends:

```
Source address:          128.99.4.123
Destination address:     128.99.4.62
Protocol:                ICMP = 1
Type:                    Address Mask Request = AM1
Code:                    0
Mask:                    0
```

128.99.4.62 should respond exactly as in the previous case.

```
Source address:          128.99.4.62
Destination address:     128.99.4.123
Protocol:                ICMP = 1
Type:                    Address Mask Reply = AM2
Code:                    0
Mask:                    255.255.252.0
```

3. A Class C Network Case (illustrating non-contiguous subnet bits)

For this case, assume that the requesting host is on class C network 192.1.127.0, has address 192.1.127.19, that there is a gateway at 192.1.127.50, and that on network an 3-bit subnet field is in use (01011000), that is, the address mask is 255.255.255.88.

The host sends the ICMP request to 255.255.255.255:

```
Source address:          192.1.127.19
Destination address:     255.255.255.255
Protocol:                ICMP = 1
Type:                    Address Mask Request = AM1
Code:                    0
Mask:                    0
```

The gateway can then respond directly to the requesting host.

```
Source address:          192.1.127.50
Destination address:     192.1.127.19
Protocol:                ICMP = 1
Type:                    Address Mask Reply = AM2
Code:                    0
Mask:                    255.255.255.88.
```

In the diskless workstation case the host sends:

```
Source address:          0.0.0.0
Destination address:     255.255.255.255
Protocol:                ICMP = 1
Type:                    Address Mask Request = AM1
Code:                    0
Mask:                    0
```

192.1.127.50 will hear the datagram, and should respond with this datagram:

```
Source address:          192.1.127.50
```

```
      Destination address:     255.255.255.255
      Protocol:                ICMP = 1
      Type:                    Address Mask Reply = AM2
      Code:                    0
      Mask:                    255.255.255.88.
```

If broadcasting is not allowed 192.1.127.19 sends:

```
      Source address:          192.1.127.19
      Destination address:     192.1.127.50
      Protocol:                ICMP = 1
      Type:                    Address Mask Request = AM1
      Code:                    0
      Mask:                    0
```

192.1.127.50 should respond exactly as in the previous case.

```
      Source address:          192.1.127.50
      Destination address:     192.1.127.19
      Protocol:                ICMP = 1
      Type:                    Address Mask Reply = AM2
      Code:                    0
      Mask:                    255.255.255.88
```

Appendix III. Glossary

Bridge

A node connected to two or more administratively indistinguishable but physically distinct subnets, that automatically forwards datagrams when necessary, but whose existence is not known to other hosts. Also called a "software repeater".

Gateway

A node connected to two or more administratively distinct networks and/or subnets, to which hosts send datagrams to be forwarded.

Host Field

The bit field in an Internet address used for denoting a specific host.

Internet

The collection of connected networks using the IP protocol.

Local Address

The rest field of the Internet address (as defined in [3]).

Network

A single Internet network (which may or may not be divided into subnets).

Network Number

The network field of the Internet address.

Subnet

One or more physical networks forming a subset of an Internet network. A subnet is explicitly identified in the Internet address.

Subnet Field

The bit field in an Internet address denoting the subnet number. The bits making up this field are not necessarily contiguous in the address.

Subnet Number

A number identifying a subnet within a network.

Appendix IV. Assigned Numbers

The following assignments are made for protocol parameters used in the support of subnets. The only assignments needed are for the Internet Control Message Protocol (ICMP) [5].

ICMP Message Types

AM1 = 17

AM2 = 18

References

[1] Mogul, J., "Internet Subnets", RFC-917, Stanford University, October 1984.

[2] Postel, J., "Multi-LAN Address Resolution", RFC-925, USC/Information Sciences Institute, October 1984.

[3] Postel, J., "Internet Protocol", RFC-791, USC/Information Sciences Institute, September 1981.

[4] Finlayson, R., T. Mann, J. Mogul, M. Theimer, "A Reverse Address Resolution Protocol", RFC-903, Stanford University, June 1984.

[5] Postel, J., "Internet Control Message Protocol", RFC-792, USC/Information Sciences Institute, September 1981.

[6] Mogul, J., "Broadcasting Internet Datagrams", RFC-919, Stanford University, October 1984.

[7] GADS, "Towards an Internet Standard Scheme for Subnetting", RFC-940, Network Information Center, SRI International, April 1985.

[8] Croft, B., and J. Gilmore, "BOOTP -- UDP Bootstrap Protocol", RFC-951, Stanford University, August 1985.

[9] Reynolds, J., and J. Postel, "Assigned Numbers", RFC-943, USC/Information Sciences Institute, April 1985.

==

Network Working Group
Request for Comments: 919

Jeffrey Mogul
Computer Science Department
Stanford University

October 1984

BROADCASTING INTERNET DATAGRAMS

Status of this Memo

We propose simple rules for broadcasting Internet datagrams on local networks that support broadcast, for addressing broadcasts, and for how gateways should handle them.

This RFC suggests a proposed protocol for the ARPA-Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Acknowledgement

This proposal is the result of discussion with several other people, especially J. Noel Chiappa and Christopher A. Kent, both of whom both pointed me at important references.

1. Introduction

The use of broadcasts, especially on high-speed local area networks, is a good base for many applications. Since broadcasting is not covered in the basic IP specification [13], there is no agreed-upon way to do it, and so protocol designers have not made use of it. (The issue has been touched upon before, e.g. [6], but has not been the subject of a standard.)

We consider here only the case of unreliable, unsequenced, possibly duplicated datagram broadcasts (for a discussion of TCP broadcasting, see [11].) Even though unreliable and limited in length, datagram broadcasts are quite useful [1].

We assume that the data link layer of the local network supports efficient broadcasting. Most common local area networks do support broadcast; for example, Ethernet [7, 5], ChaosNet [10], token ring networks [2], etc.

We do not assume, however, that broadcasts are reliably delivered. (One might consider providing a reliable broadcast protocol as a layer above IP.) It is quite expensive to guarantee delivery of broadcasts; instead, what we assume is that a host will receive most of the broadcasts that are sent. This is important to avoid excessive use of broadcasts; since every host on the network devotes at least some effort to every broadcast, they are costly.

When a datagram is broadcast, it imposes a cost on every host that hears it. Therefore, broadcasting should not be used indiscriminately, but rather only when it is the best solution to a problem.

Note: some organizations have divided their IP networks into subnets, for which a standard [8] has been proposed. This RFC does not cover the numerous complications arising from the interactions between subnets and broadcasting; see [9] for a complete discussion.

2. Terminology

Because broadcasting depends on the specific data link layer in use on a local network, we must discuss it with reference to both physical networks and logical networks.

The terms we will use in referring to physical networks are, from the point of view of the host sending or forwarding a broadcast:

Local Hardware Network

The physical link to which the host is attached.

Remote Hardware Network

A physical network which is separated from the host by at least one gateway.

Collection of Hardware Networks

A set of hardware networks (transitively) connected by gateways.

The IP world includes several kinds of logical network. To avoid ambiguity, we will use the following terms:

Internet

The DARPA Internet collection of IP networks.

IP Network

One or a collection of several hardware networks that have one specific IP network number.

3. Why Broadcast?

Broadcasts are useful when a host needs to find information without knowing exactly what other host can supply it, or when a host wants to provide information to a large set of hosts in a timely manner.

When a host needs information that one or more of its neighbors might have, it could have a list of neighbors to ask, or it could poll all of its possible neighbors until one responds. Use of a wired-in list creates obvious network management problems (early binding is inflexible). On the other hand, asking all of one's neighbors is slow if one must generate plausible host addresses, and try them until one works. On the ARPANET, for example, there are roughly 65 thousand plausible host numbers. Most IP implementations have used wired-in lists (for example, addresses of "Prime" gateways.)

Fortunately, broadcasting provides a fast and simple way for a host to reach all of its neighbors.

A host might also use a broadcast to provide all of its neighbors with some information; for example, a gateway might announce its presence to other gateways.

One way to view broadcasting is as an imperfect substitute for multicasting, the sending of messages to a subset of the hosts on a network. In practice, broadcasts are usually used where multicasts are what is wanted; packets are broadcast at the hardware level, but filtering software in the receiving hosts gives the effect of multicasting.

For more examples of broadcast applications, see [1, 3].

4. Broadcast Classes

There are several classes of IP broadcasting:

- Single-destination datagram broadcast on the local IP net: A datagrams is destined for a specific IP host, but the sending host broadcasts it at the data link layer, perhaps to avoid having to do routing. Since this is not an IP broadcast, the IP layer is not involved, except that a host should discard datagrams not meant for it without becoming flustered (i.e., printing an error message).

- Broadcast to all hosts on the local IP net: A distinguished value for the host-number part of the IP address denotes broadcast instead of a specific host. The receiving IP layer must be able to recognize this address as well as its own.

  However, it might still be useful to distinguish at higher levels between broadcasts and non-broadcasts, especially in gateways. This is the most useful case of broadcast; it allows a host to discover gateways without wired-in tables, it is the basis for address resolution protocols, and it is also useful for accessing such utilities as name servers, time servers, etc., without requiring wired-in addresses.

- Broadcast to all hosts on a remote IP network: It is occasionally useful to send a broadcast to all hosts on a non-local network; for example, to find the latest version of a hostname database, to bootload a host on an IP network without a bootserver, or to monitor the timeservers on the IP network. This case is the same as local-network broadcasts; the datagram is routed by normal mechanisms until it reaches a gateway attached to the destination IP network, at which point it is broadcast. This class of broadcasting is also known as "directed broadcasting", or quaintly as sending a "letter bomb" [1].

- Broadcast to the entire Internet: This is probably not useful, and almost certainly not desirable.

For reasons of performance or security, a gateway may choose not to forward broadcasts; especially, it may be a good idea to ban broadcasts into or out of an autonomous group of networks.

5. Broadcast Methods

A host's IP receiving layer must be modified to support broadcasting. In the absence of broadcasting, a host determines if it is the recipient of a datagram by matching the destination address against all of its IP addresses. With broadcasting, a host must compare the destination address not only against the host's addresses, but also against the possible broadcast addresses for that host.

The problem of how best to send a broadcast has been extensively discussed [1, 3, 4, 14, 15]. Since we assume that the problem has already been solved at the data link layer, an IP host wishing to send either a local broadcast or a directed broadcast need only specify the appropriate destination address and send the datagram as usual. Any sophisticated algorithms need only reside in gateways.

6. Gateways and Broadcasts

Most of the complexity in supporting broadcasts lies in gateways. If a gateway receives a directed broadcast for a network to which it is not connected, it simply forwards it using the usual mechanism. Otherwise, it must do some additional work.

When a gateway receives a local broadcast datagram, there are several things it might have to do with it. The situation is unambiguous, but without due care it is possible to create infinite loops.

The appropriate action to take on receipt of a broadcast datagram depends on several things: the subnet it was received on, the destination network, and the addresses of the gateway.

- The primary rule for avoiding loops is "never broadcast a datagram on the hardware network it was received on". It is not sufficient simply to avoid repeating datagrams that a gateway has heard from itself; this still allows loops if there are several gateways on a hardware network.

- If the datagram is received on the hardware network to which it is addressed, then it should not be forwarded. However, the gateway should consider itself to be a destination of the datagram (for example, it might be a routing table update.)

- Otherwise, if the datagram is addressed to a hardware network to which the gateway is connected, it should be sent as a (data link layer) broadcast on that network. Again, the gateway should consider itself a destination of the datagram.

- Otherwise, the gateway should use its normal routing procedure to choose a subsequent gateway, and send the datagram along to it.

7. Broadcast IP Addressing - Proposed Standards

If different IP implementations are to be compatible, there must be a distinguished number to denote "all hosts".

Since the local network layer can always map an IP address into data link layer address, the choice of an IP "broadcast host number" is somewhat arbitrary. For simplicity, it should be one not likely to be assigned to a real host. The number whose bits are all ones has this property; this assignment was first proposed in [6]. In the few cases where a host has been assigned an address with a host-number part of all ones, it does not seem onerous to require renumbering.

The address 255.255.255.255 denotes a broadcast on a local hardware network, which must not be forwarded. This address may be used, for example, by hosts that do not know their network number and are asking some server for it.

Thus, a host on net 36, for example, may:

- broadcast to all of its immediate neighbors by using 255.255.255.255

- broadcast to all of net 36 by using 36.255.255.255

(Note that unless the network has been broken up into subnets, these two methods have identical effects.)

If the use of "all ones" in a field of an IP address means "broadcast", using "all zeros" could be viewed as meaning "unspecified". There is probably no reason for such addresses to appear anywhere but as the source address of an ICMP Information Request datagram. However, as a notational convention, we refer to networks (as opposed to hosts) by using addresses with zero fields. For example, 36.0.0.0 means "network number 36" while 36.255.255.255 means "all hosts on network number 36".

7.1. ARP Servers and Broadcasts

The Address Resolution Protocol (ARP) described in [12] can, if incorrectly implemented, cause problems when broadcasts are used on a network where not all hosts share an understanding of what a broadcast address is. The temptation exists to modify the ARP server so that it provides the mapping between an IP broadcast address and the hardware broadcast address.

This temptation must be resisted. An ARP server should never respond to a request whose target is a broadcast address. Such a request can only come from a host that does not recognize the broadcast address as such, and so honoring it would almost certainly lead to a forwarding loop. If there are N such hosts on the physical network that do not recognize this address as a broadcast, then a datagram sent with a Time-To-Live of T could potentially give rise to T**N spurious re-broadcasts.

8. References

1. David Reeves Boggs. Internet Broadcasting. Ph.D. Th., Stanford University, January 1982.

2. D.D. Clark, K.T. Pogran, and D.P. Reed. "An Introduction to Local Area Networks". Proc. IEEE 66, 11, pp1497-1516, 1978.

3. Yogan Kantilal Dalal. Broadcast Protocols in Packet Switched Computer Networks. Ph.D. Th., Stanford University, April 1977.

4. Yogan K. Dalal and Robert M. Metcalfe. "Reverse Path Forwarding of Broadcast Packets". Comm. ACM 21, 12, pp1040-1048, December 1978.

5. The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specifications. Version 1.0, Digital Equipment Corporation, Intel, Xerox, September 1980.

6. Robert Gurwitz and Robert Hinden. IP - Local Area Network Addressing Issues. IEN-212, Bolt Beranek and Newman, September 1982.

7. R.M. Metcalfe and D.R. Boggs. "Ethernet: Distributed Packet Switching for Local Computer Networks". Comm. ACM 19, 7, pp395-404, July 1976. Also CSL-75-7, Xerox Palo Alto Research Center, reprinted in CSL-80-2.

8. Jeffrey Mogul. Internet Subnets. RFC-917, Stanford University, October 1984.

9. Jeffrey Mogul. Broadcasting Internet Packets in the Presence of Subnets. RFC-922, Stanford University, October 1984.

10. David A. Moon. Chaosnet. A.I. Memo 628, Massachusetts Institute of Technology Artificial Intelligence Laboratory, June 1981.

11. William W. Plummer. Internet Broadcast Protocols. IEN-10, Bolt Beranek and Newman, March 1977.

12. David Plummer. An Ethernet Address Resolution Protocol. RFC-826, Symbolics, September 1982.

13. Jon Postel. Internet Protocol. RFC 791, ISI, September 1981.

14. David W. Wall. Mechanisms for Broadcast and Selective Broadcast. Ph.D. Th., Stanford University, June 1980.

15. David W. Wall and Susan S. Owicki. Center-based Broadcasting. Computer Systems Lab Technical Report TR189, Stanford University, June 1980.

==

Network Working Group   Jeffrey Mogul
Request for Comments: 922   Computer Science Department
                                                           Stanford University
                                                                  October 1984

BROADCASTING INTERNET DATAGRAMS IN THE PRESENCE OF SUBNETS

Status of this Memo

We propose simple rules for broadcasting Internet datagrams on local networks that support broadcast, for addressing broadcasts, and for how gateways should handle them.

This RFC suggests a proposed protocol for the ARPA-Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Acknowledgement

This proposal here is the result of discussion with several other people, especially J. Noel Chiappa and Christopher A. Kent, both of whom both pointed me at important references.

1. Introduction

The use of broadcasts, especially on high-speed local area networks, is a good base for many applications. Since broadcasting is not covered in the basic IP specification [12], there is no agreed-upon way to do it, and so protocol designers have not made use of it. (The issue has been touched upon before, e.g. [6], but has not been the subject of a standard.)

We consider here only the case of unreliable, unsequenced, possibly duplicated datagram broadcasts (for a discussion of TCP broadcasting, see [10].) Even though unreliable and limited in length, datagram broadcasts are quite useful [1].

We assume that the data link layer of the local network supports efficient broadcasting. Most common local area networks do support broadcast; for example, Ethernet [7, 5], ChaosNet [9], token ring networks [2], etc.

We do not assume, however, that broadcasts are reliably delivered. (One might consider providing a reliable datagram broadcast protocol as a layer above IP.) It is quite expensive to guarantee delivery of broadcasts; instead, what we assume is that a host will receive most of the broadcasts that are sent. This is important to avoid excessive use of broadcasts; since every host on the network devotes at least some effort to every broadcast, they are costly.

When a datagram is broadcast, it imposes a cost on every host that hears it. Therefore, broadcasting should not be used indiscriminately, but rather only when it is the best solution to a problem.

2. Terminology

Because broadcasting depends on the specific data link layer in use on a local network, we must discuss it with reference to both physical networks and logical networks.

The terms we will use in referring to physical networks are, from the point of view of the host sending or forwarding a broadcast:

Local Hardware Network

The physical link to which the host is attached.

Remote Hardware Network

A physical network which is separated from the host by at least one gateway.

Collection of Hardware Networks

A set of hardware networks (transitively) connected by gateways.

The IP world includes several kinds of logical network. To avoid ambiguity, we will use the following terms:

Internet

The DARPA Internet collection of IP networks.

IP Network

One or a collection of several hardware networks that have one specific IP network number.

Subnet

A single member of the collection of hardware networks that compose an IP network. Host addresses on a given subnet share an IP network number with hosts on all other subnets of that IP network, but the local-address part is divided into subnet-number and host-number fields to indicate which subnet a host is on. We do not assume a particular division of the local-address part; this could vary from network to network.

The introduction of a subnet level in the addressing hierarchy is at variance with the IP specification [12], but as the use of addressable subnets proliferates it is obvious that a broadcasting scheme should support subnetting. For more on subnets, see [8].

In this paper, the term "host address" refers to the host-on-subnet address field of a subnetted IP network, or the host-part field otherwise.

An IP network may consist of a single hardware network or a collection of subnets; from the point of view of a host on another IP network, it should not matter.

3. Why Broadcast?

Broadcasts are useful when a host needs to find information without knowing exactly what other host can supply it, or when a host wants to provide information to a large set of hosts in a timely manner.

When a host needs information that one or more of its neighbors might have, it could have a list of neighbors to ask, or it could poll all of its possible neighbors until one responds. Use of a wired-in list creates obvious network management problems (early binding is inflexible). On the other hand, asking all of one's neighbors is slow if one must generate plausible host addresses, and try them until one works. On the ARPANET, for example, there are roughly 65 thousand plausible host numbers. Most IP implementations have used wired-in lists (for example, addresses of "Prime" gateways.) Fortunately, broadcasting provides a fast and simple way for a host to reach all of its neighbors.

A host might also use a broadcast to provide all of its neighbors with some information; for example, a gateway might announce its presence to other gateways.

One way to view broadcasting is as an imperfect substitute for multicasting, the sending of messages to a subset of the hosts on a network. In practice, broadcasts are usually used where multicasts are what is wanted; datagrams are broadcast at the hardware level, but filtering software in the receiving hosts gives the effect of multicasting.

For more examples of broadcast applications, see [1, 3].

4. Broadcast Classes

There are several classes of IP broadcasting:

- Single-destination datagrams broadcast on the local hardware net: A datagram is destined for a specific IP host, but the sending host broadcasts it at the data link layer, perhaps to avoid having to do routing. Since this is not an IP broadcast, the IP layer is not involved, except that a host should discard datagram not meant for it without becoming flustered (i.e., printing an error message).

- Broadcast to all hosts on the local hardware net: A distinguished value for the host-number part of the IP address denotes broadcast instead of a specific host. The receiving IP layer must be able to recognize this address as well as its own. However, it might still be useful to distinguish at higher levels between broadcasts and non-broadcasts, especially in gateways. This is the most useful case of broadcast; it allows a host to discover gateways without wired-in tables, it is the basis for address resolution protocols, and it is also useful for accessing such utilities as name servers, time servers, etc., without requiring wired-in addresses.

- Broadcast to all hosts on a remote hardware network: It is occasionally useful to send a broadcast to all hosts on a non-local network; for example, to find the latest version of a hostname database, to bootload a host on a subnet without a bootserver, or to monitor the timeservers on the subnet. This case is the same as local-network broadcasts; the datagram is routed by normal mechanisms until it reaches a gateway attached to the destination hardware network, at which point it is broadcast. This class of broadcasting is also known as "directed broadcasting", or quaintly as sending a "letter bomb" [1].

- Broadcast to all hosts on a subnetted IP network (Multi-subnet broadcasts): A distinguished value for the subnet-number part of the IP address is used to denote "all subnets". Broadcasts to all hosts of a remote subnetted IP network are done just as directed broadcasts to a single subnet.

- Broadcast to the entire Internet: This is probably not useful, and almost certainly not desirable.

For reasons of performance or security, a gateway may choose not to forward broadcasts; especially, it may be a good idea to ban broadcasts into or out of an autonomous group of networks.

5. Broadcast Methods

A host's IP receiving layer must be modified to support broadcasting. In the absence of broadcasting, a host determines if it is the recipient of a datagram by matching the destination address against all of its IP addresses. With broadcasting, a host must compare the destination address not only against the host's addresses, but also against the possible broadcast addresses for that host.

The problem of how best to send a broadcast has been extensively discussed [1, 3, 4, 13, 14]. Since we assume that the problem has already been solved at the data link layer, an IP host wishing to send either a local broadcast or a directed broadcast need only specify the appropriate destination address and send the datagram as usual. Any sophisticated algorithms need only reside in gateways.

The problem of broadcasting to all hosts on a subnetted IP network is apparently somewhat harder. However, even in this case it turns out that the best known algorithms require no additional complexity in non-gateway hosts. A good broadcast method will meet these additional criteria:

- No modification of the IP datagram format.

- Reasonable efficiency in terms of the number of excess copies generated and the cost of paths chosen.

- Minimization of gateway modification, in both code and data space.

- High likelihood of delivery.

The algorithm that appears best is the Reverse Path Forwarding (RPF) method [4]. While RPF is suboptimal in cost and reliability, it is quite good, and is extremely simple to implement, requiring no additional data space in a gateway.

6. Gateways and Broadcasts

Most of the complexity in supporting broadcasts lies in gateways. If a gateway receives a directed broadcast for a network to which it is not connected, it simply forwards it using the usual mechanism. Otherwise, it must do some additional work.

6.1. Local Broadcasts

When a gateway receives a local broadcast datagram, there are several things it might have to do with it. The situation is unambiguous, but without due care it is possible to create infinite loops.

The appropriate action to take on receipt of a broadcast datagram depends on several things: the subnet it was received on, the destination network, and the addresses of the gateway.

- The primary rule for avoiding loops is "never broadcast a datagram on the hardware network it was received on". It is not sufficient simply to avoid repeating datagram that a gateway has heard from itself; this still allows loops if there are several gateways on a hardware network.

- If the datagram is received on the hardware network to which it is addressed, then it should not be forwarded. However, the gateway should consider itself to be a destination of the datagram (for example, it might be a routing table update.)

- Otherwise, if the datagram is addressed to a hardware network to which the gateway is connected, it should be sent as a (data link layer) broadcast on that network. Again, the gateway should consider itself a destination of the datagram.

- Otherwise, the gateway should use its normal routing procedure to choose a subsequent gateway, and send the datagram along to it.

6.2. Multi-subnet broadcasts

When a gateway receives a broadcast meant for all subnets of an IP network, it must use the Reverse Path Forwarding algorithm to decide what to do. The method is simple: the gateway should forward copies of the datagram along all connected links, if and only if the datagram arrived on the link which is part of the best route between the gateway and the source of the datagram. Otherwise, the datagram should be discarded.

This algorithm may be improved if some or all of the gateways exchange among themselves additional information; this can be done transparently from the point of view of other hosts and even other gateways. See [4, 3] for details.

6.3. Pseudo-Algol Routing Algorithm

This is a pseudo-Algol description of the routing algorithm a gateway should use. The algorithm is shown in figure 1. Some definitions are:

RouteLink(host)

> A function taking a host address as a parameter and returning the first-hop link from the gateway to the host.

RouteHost(host)

> As above but returns the first-hop host address.

ResolveAddress(host)

> Returns the hardware address for an IP host.

IncomingLink

> The link on which the packet arrived.

OutgoingLinkSet

> The set of links on which the packet should be sent.

OutgoingHardwareHost

> The hardware host address to send the packet to.

Destination.host

> The host-part of the destination address.

Destination.subnet

> The subnet-part of the destination address.

Destination.ipnet

> The IP-network-part of the destination address.

```
BEGIN
   IF Destination.ipnet IN AllLinks THEN
      BEGIN
         IF IsSubnetted(Destination.ipnet) THEN
            BEGIN
               IF Destination.subnet = BroadcastSubnet THEN
                  BEGIN      /* use Reverse Path Forwarding algorithm */
                     IF IncomingLink = RouteLink(Source) THEN
```

```
                        BEGIN IF Destination.host = BroadcastHost THEN
                              OutgoingLinkSet <- AllLinks -
                           IncomingLink;
                           OutgoingHost <- BroadcastHost;
                           Examine packet for possible internal use;
                        END
                     ELSE  /* duplicate from another gateway, discard */
                        Discard;
                  END
               ELSE
                  IF Destination.subnet = IncomingLink.subnet THEN
                     BEGIN           /* forwarding would cause a loop */
                        IF Destination.host = BroadcastHost THEN
                           Examine packet for possible internal use;
                        Discard;
                     END
                  ELSE BEGIN    /* forward to (possibly local) subnet */
                        OutgoingLinkSet <- RouteLink(Destination);
                        OutgoingHost <- RouteHost(Destination);
                     END
            END
         ELSE BEGIN         /* destined for one of our local networks */
               IF Destination.ipnet = IncomingLink.ipnet THEN
                  BEGIN              /* forwarding would cause a loop */
                     IF Destination.host = BroadcastHost THEN
                        Examine packet for possible internal use;
                     Discard;
                  END
               ELSE BEGIN                     /* might be a broadcast */
                     OutgoingLinkSet <- RouteLink(Destination);
                     OutgoingHost <- RouteHost(Destination);
                  END
            END
      END
   ELSE BEGIN                    /* forward to a non-local IP network */
         OutgoingLinkSet <- RouteLink(Destination);
         OutgoingHost <- RouteHost(Destination);
      END
   OutgoingHardwareHost <- ResolveAddress(OutgoingHost);
END
```

Figure 1: Pseudo-Algol algorithm for routing broadcasts by gateways

7. Broadcast IP Addressing - Conventions

If different IP implementations are to be compatible, there must be convention distinguished number to denote "all hosts" and "all subnets".

Since the local network layer can always map an IP address into data link layer address, the choice of an IP "broadcast host number" is somewhat arbitrary. For simplicity, it should be one not likely to be assigned to a real host. The number whose bits are all ones has this property; this assignment was first proposed in [6]. In the few cases where a host has been assigned an address with a host-number part of all ones, it does not seem onerous to require renumbering.

The "all subnets" number is also all ones; this means that a host wishing to broadcast to all hosts on a remote IP network need not know how the destination address is divided up into subnet and host fields, or if it is even divided at all. For example, 36.255.255.255 may denote all the hosts on a single hardware network, or all the hosts on a subnetted IP network with 1 byte of subnet field and 2 bytes of host field, or any other possible division.

The address 255.255.255.255 denotes a broadcast on a local hardware network that must not be forwarded. This address may be used, for example, by hosts that do not know their network number and are asking some server for it.

Thus, a host on net 36, for example, may:

- broadcast to all of its immediate neighbors by using 255.255.255.255

- broadcast to all of net 36 by using 36.255.255.255

without knowing if the net is subnetted; if it is not, then both addresses have the same effect. A robust application might try the former address, and if no response is received, then try the latter. See [1] for a discussion of such "expanding ring search" techniques.

If the use of "all ones" in a field of an IP address means "broadcast", using "all zeros" could be viewed as meaning "unspecified". There is probably no reason for such addresses to appear anywhere but as the source address of an ICMP Information Request datagram. However, as a notational convention, we refer to networks (as opposed to hosts) by using addresses with zero fields. For example, 36.0.0.0 means "network number 36" while 36.255.255.255 means "all hosts on network number 36".

7.1. ARP Servers and Broadcasts

The Address Resolution Protocol (ARP) described in [11] can, if incorrectly implemented, cause problems when broadcasts are used on a network where not all hosts share an understanding of what a broadcast address is. The temptation exists to modify the ARP server so that it provides the mapping between an IP broadcast address and the hardware broadcast address.

This temptation must be resisted. An ARP server should never respond to a request whose target is a broadcast address. Such a request can only come from a host that does not recognize the broadcast address as such, and so honoring it would almost certainly lead to a forwarding loop. If there are N such hosts on the physical network that do not recognize this address as a broadcast, then a datagram sent with a Time-To-Live of T could potentially give rise to T**N spurious re-broadcasts.

8. References

1. David Reeves Boggs. Internet Broadcasting. Ph.D. Th., Stanford University, January 1982.

2. D.D. Clark, K.T. Pogran, and D.P. Reed. "An Introduction to Local Area Networks". Proc. IEEE 66, 11, pp1497-1516, November 1978.

3. Yogan Kantilal Dalal. Broadcast Protocols in Packet Switched Computer Networks. Ph.D. Th., Stanford University, April 1977.

4. Yogan K. Dalal and Robert M. Metcalfe. "Reverse Path Forwarding of Broadcast Packets". Comm. ACM 21, 12, pp1040-1048, December 1978.

5. The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specifications. Version 1.0, Digital Equipment Corporation, Intel, Xerox, September 1980.

6. Robert Gurwitz and Robert Hinden. IP - Local Area Network Addressing Issues. IEN-212, BBN, September 1982.

7. R.M. Metcalfe and D.R. Boggs. "Ethernet: Distributed Packet Switching for Local Computer Networks". Comm. ACM 19, 7, pp395-404, July 1976. Also CSL-75-7, Xerox Palo Alto Research Center, reprinted in CSL-80-2.

8. Jeffrey Mogul. Internet Subnets. RFC-917, Stanford University, October 1984.

9. David A. Moon. Chaosnet. A.I. Memo 628, Massachusetts Institute of Technology Artificial Intelligence Laboratory, June 1981.

10. William W. Plummer. Internet Broadcast Protocols. IEN-10, BBN, March 1977.

11. David Plummer. An Ethernet Address Resolution Protocol. RFC-826, Symbolics, September 1982.

12. Jon Postel. Internet Protocol. RFC-791, ISI, September 1981.

13. David W. Wall. Mechanisms for Broadcast and Selective Broadcast. Ph.D. Th., Stanford University, June 1980.

14. David W. Wall and Susan S. Owicki. Center-based Broadcasting. Computer Systems Lab Technical Report TR189, Stanford University, June 1980.

==

Network Working Group J. Postel
Request for Comments: 792 ISI
September 1981

Updates: RFCs 777, 760
Updates: IENs 109, 128

INTERNET CONTROL MESSAGE PROTOCOL

DARPA INTERNET PROGRAM
PROTOCOL SPECIFICATION

Introduction

The Internet Protocol (IP) [1] is used for host-to-host datagram service in a system of interconnected networks called the Catenet [2]. The network connecting devices are called Gateways. These gateways communicate between themselves for control purposes via a Gateway to Gateway Protocol (GGP) [3,4]. Occasionally a gateway or destination host will communicate with a source host, for example, to report an error in datagram processing. For such purposes this protocol, the Internet Control Message Protocol (ICMP), is used. ICMP, uses the basic support of IP as if it were a higher level protocol, however, ICMP is actually an integral part of IP, and must be implemented by every IP module.

ICMP messages are sent in several situations: for example, when a datagram cannot reach its destination, when the gateway does not have the buffering capacity to forward a datagram, and when the gateway can direct the host to send traffic on a shorter route.

The Internet Protocol is not designed to be absolutely reliable. The purpose of these control messages is to provide feedback about problems in the communication environment, not to make IP reliable. There are still no guarantees that a datagram will be delivered or a control message will be returned. Some datagrams may still be undelivered without any report of their loss. The higher level protocols that use IP must implement their own reliability procedures if reliable communication is required.

The ICMP messages typically report errors in the processing of datagrams. To avoid the infinite regress of messages about messages etc., no ICMP messages are sent about ICMP messages. Also ICMP messages are only sent about errors in handling fragment zero of fragemented datagrams. (Fragment zero has the fragment offeset equal zero).

ICMP messages are sent using the basic IP header. The first octet of the data portion of the datagram is a ICMP type field; the value of this field determines the format of the remaining data. Any field labeled "unused" is reserved for later extensions and must be zero when sent, but receivers should not use these fields (except to include them in the checksum). Unless otherwise noted under the individual format descriptions, the values of the internet header fields are as follows:

Version

4

IHL

Internet header length in 32-bit words.

Type of Service

0

Total Length

Length of internet header and data in octets.

Identification, Flags, Fragment Offset

Used in fragmentation, see [1].

Time to Live

Time to live in seconds; as this field is decremented at each machine in which the datagram is processed, the value in this field should be at least as great as the number of gateways which this datagram will traverse.

Protocol

ICMP = 1

Header Checksum

The 16 bit one's complement of the one's complement sum of all 16 bit words in the header. For computing the checksum, the checksum field should be zero. This checksum may be replaced in the future.

Source Address

The address of the gateway or host that composes the ICMP message. Unless otherwise noted, this can be any of a gateway's addresses.

Destination Address

The address of the gateway or host to which the message should be sent.

Destination Unreachable Message

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             unused                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Fields:

Destination Address

The source network and address from the original datagram's data.

ICMP Fields:

Type

3

Code

0 = net unreachable;

1 = host unreachable;

2 = protocol unreachable;

3 = port unreachable;

4 = fragmentation needed and DF set;

5 = source route failed.

Checksum

The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

Internet Header + 64 bits of Data Datagram

The internet header plus the first 64 bits of the original datagram's data. This data is used by the host to match the message to the appropriate process. If a higher level protocol uses port numbers, they are assumed to be in the first 64 data bits of the original datagram's data.

Description

If, according to the information in the gateway's routing tables, the network specified in the internet destination field of a datagram is unreachable, e.g., the distance to the network is infinity, the gateway may send a destination unreachable message to the internet source host of the datagram. In addition, in some networks, the gateway may be able to determine if the internet destination host is unreachable. Gateways in these networks may send destination unreachable messages to the source host when the destination host is unreachable.

If, in the destination host, the IP module cannot deliver the datagram because the indicated protocol module or process port is not active, the destination host may send a destination unreachable message to the source host.

Another case is when a datagram must be fragmented to be forwarded by a gateway yet the Don't Fragment flag is on. In this case the gateway must discard the datagram and may return a destination unreachable message.

Codes 0, 1, 4, and 5 may be received from a gateway. Codes 2 and 3 may be received from a host.

Time Exceeded Message

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             unused                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Fields:

Destination Address

The source network and address from the original datagram's data.

ICMP Fields:

Type

11

Code

0 = time to live exceeded in transit;

1 = fragment reassembly time exceeded.

Checksum

The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

Internet Header + 64 bits of Data Datagram

The internet header plus the first 64 bits of the original datagram's data. This data is used by the host to match the message to the appropriate process. If a higher level protocol uses port numbers, they are assumed to be in the first 64 data bits of the original datagram's data.

Description

If the gateway processing a datagram finds the time to live field is zero it must discard the datagram. The gateway may also notify the source host via the time exceeded message.

If a host reassembling a fragmented datagram cannot complete the reassembly due to missing fragments within its time limit it discards the datagram, and it may send a time exceeded message.

If fragment zero is not available then no time exceeded need be sent at all.

Code 0 may be received from a gateway. Code 1 may be received from a host.

Parameter Problem Message

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Pointer    |                   unused                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Fields:

Destination Address

The source network and address from the original datagram's data.

ICMP Fields:

Type

12

Code

0 = pointer indicates the error.

Checksum

The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

Pointer

If code = 0, identifies the octet where an error was detected.

Internet Header + 64 bits of Data Datagram

The internet header plus the first 64 bits of the original datagram's data. This data is used by the host to match the message to the appropriate process. If a higher level protocol uses port numbers, they are assumed to be in the first 64 data bits of the original datagram's data.

Description

If the gateway or host processing a datagram finds a problem with the header parameters such that it cannot complete processing the datagram it must discard the datagram. One potential source of such a problem is with incorrect arguments in an option. The gateway or host may also notify the source host via the parameter problem message. This message is only sent if the error caused the datagram to be discarded.

The pointer identifies the octet of the original datagram's header where the error was detected (it may be in the middle of an option). For example, 1 indicates something is wrong with the Type of Service, and (if there are options present) 20 indicates something is wrong with the type code of the first option.

Code 0 may be received from a gateway or a host.

Source Quench Message

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             unused                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Fields:

Destination Address

The source network and address of the original datagram's data.

ICMP Fields:

Type

4

Code

0

Checksum

The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

Internet Header + 64 bits of Data Datagram

The internet header plus the first 64 bits of the original datagram's data. This data is used by the host to match the message to the appropriate process. If a higher level protocol uses port numbers, they are assumed to be in the first 64 data bits of the original datagram's data.

Description

A gateway may discard internet datagrams if it does not have the buffer space needed to queue the datagrams for output to the next network on the route to the destination network. If a gateway discards a datagram, it may send a source quench message to the internet source host of the datagram. A destination host may also send a source quench message if datagrams arrive too fast to be processed. The source quench message is a request to the host to cut back the rate at which it is sending traffic to the internet destination. The gateway may send a source quench message for every message that it discards. On receipt of a source quench message, the source host should cut back the rate at which it is sending traffic to the specified destination until it no longer receives source quench messages from the gateway. The source host can then gradually increase the rate at which it sends traffic to the destination until it again receives source quench messages.

The gateway or host may send the source quench message when it approaches its capacity limit rather than waiting until the capacity is exceeded. This means that the data datagram which triggered the source quench message may be delivered.

Code 0 may be received from a gateway or a host.

Redirect Message

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |     Code      |          Checksum             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                 Gateway Internet Address                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      Internet Header + 64 bits of Original Data Datagram      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Fields:

Destination Address

The source network and address of the original datagram's data.

ICMP Fields:

Type

5

Code

0 = Redirect datagrams for the Network.

1 = Redirect datagrams for the Host.

2 = Redirect datagrams for the Type of Service and Network.

3 = Redirect datagrams for the Type of Service and Host.

Checksum

The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

Gateway Internet Address

Address of the gateway to which traffic for the network specified in the internet destination network field of the original datagram's data should be sent.

Internet Header + 64 bits of Data Datagram

The internet header plus the first 64 bits of the original datagram's data. This data is used by the host to match the message to the appropriate process. If a higher level protocol uses port numbers, they are assumed to be in the first 64 data bits of the original datagram's data.

Description

The gateway sends a redirect message to a host in the following situation. A gateway, G1, receives an internet datagram from a host on a network to which the gateway is attached. The gateway, G1, checks its routing table and obtains the address of the next gateway, G2, on the route to the datagram's internet destination network, X. If G2 and the host identified by the internet source address of the datagram are on the same network, a redirect message is sent to the host. The redirect message advises the host to send its traffic for network X directly to gateway G2 as this is a shorter path to the destination. The gateway forwards the original datagram's data to its internet destination.

For datagrams with the IP source route options and the gateway address in the destination address field, a redirect message is not sent even if there is a better route to the ultimate destination than the next address in the source route.

Codes 0, 1, 2, and 3 may be received from a gateway.

Echo or Echo Reply Message

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |     Code      |          Checksum             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           Identifier          |        Sequence Number        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Data ...
   +-+-+-+-+-
```

IP Fields:

Addresses

The address of the source in an echo message will be the destination of the echo reply message. To form an echo reply message, the source and destination addresses are simply reversed, the type code changed to 0, and the checksum recomputed.

IP Fields:

Type

8 for echo message;

0 for echo reply message.

Code

0

Checksum

The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum , the checksum field should be zero. If the total length is odd, the received data is padded with one octet of zeros for computing the checksum. This checksum may be replaced in the future.

Identifier

If code = 0, an identifier to aid in matching echos and replies, may be zero.

Sequence Number

If code = 0, a sequence number to aid in matching echos and replies, may be zero.

Description

The data received in the echo message must be returned in the echo reply message.

The identifier and sequence number may be used by the echo sender to aid in matching the replies with the echo requests. For example, the identifier might be used like a port in TCP or UDP to identify a session, and the sequence number might be incremented on each echo request sent. The echoer returns these same values in the echo reply.

Code 0 may be received from a gateway or a host.

Timestamp or Timestamp Reply Message

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |      Code     |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Identifier          |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Originate Timestamp                                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Receive Timestamp                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Transmit Timestamp                                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Fields:

Addresses

The address of the source in a timestamp message will be the destination of the timestamp reply message. To form a timestamp reply message, the source and destination addresses are simply reversed, the type code changed to 14, and the checksum recomputed.

IP Fields:

Type

13 for timestamp message;

14 for timestamp reply message.

Code

0

Checksum

The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

Identifier

If code = 0, an identifier to aid in matching timestamp and replies, may be zero.

Sequence Number

If code = 0, a sequence number to aid in matching timestamp and replies, may be zero.

Description

The data received (a timestamp) in the message is returned in the reply together with an additional timestamp. The timestamp is 32 bits of milliseconds since midnight UT. One use of these timestamps is described by Mills [5].

The Originate Timestamp is the time the sender last touched the message before sending it, the Receive Timestamp is the time the echoer first touched it on receipt, and the Transmit Timestamp is the time the echoer last touched the message on sending it.

If the time is not available in miliseconds or cannot be provided with respect to midnight UT then any time can be inserted in a timestamp provided the high order bit of the timestamp is also set to indicate this non-standard value.

The identifier and sequence number may be used by the echo sender to aid in matching the replies with the requests. For example, the identifier might be used like a port in TCP or UDP to identify a session, and the sequence number might be incremented on each request sent. The destination returns these same values in the reply.

Code 0 may be received from a gateway or a host.

Information Request or Information Reply Message

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |      Code     |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Identifier          |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Fields:

Addresses

The address of the source in a information request message will be the destination of the information reply message. To form a information reply message, the source and destination addresses are simply reversed, the type code changed to 16, and the checksum recomputed.

IP Fields:

Type

15 for information request message;

16 for information reply message.

Code

0

Checksum

The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

Identifier

If code = 0, an identifier to aid in matching request and replies, may be zero.

Sequence Number

If code = 0, a sequence number to aid in matching request and replies, may be zero.

Description

This message may be sent with the source network in the IP header source and destination address fields zero (which means "this" network). The replying IP module should send the reply with the addresses fully specified. This message is a way for a host to find out the number of the network it is on.

The identifier and sequence number may be used by the echo sender to aid in matching the replies with the requests. For example, the identifier might be used like a port in TCP or UDP to identify a session, and the sequence number might be incremented on each request sent. The destination returns these same values in the reply.

Code 0 may be received from a gateway or a host.

Summary of Message Types

0 Echo Reply

3 Destination Unreachable

4 Source Quench

5 Redirect

8 Echo

11 Time Exceeded

12 Parameter Problem

13 Timestamp

14 Timestamp Reply

15 Information Request

16 Information Reply

References

[1] Postel, J. (ed.), "Internet Protocol - DARPA Internet Program Protocol Specification," RFC 791, USC/Information Sciences Institute, September 1981.

[2] Cerf, V., "The Catenet Model for Internetworking," IEN 48,

Information Processing Techniques Office, Defense Advanced Research Projects Agency, July 1978.

[3] Strazisar, V., "Gateway Routing: An Implementation Specification", IEN 30, Bolt Beranek and Newman, April 1979.

[4] Strazisar, V., "How to Build a Gateway", IEN 109, Bolt Beranek and Newman, August 1979.

[5] Mills, D., "DCNET Internet Clock Service," RFC 778, COMSAT Laboratories, April 1981.

==

Network Working Group
Request for Comments: 1112
Obsoletes: RFCs 988, 1054
S. Deering
Stanford University
August 1989

Host Extensions for IP Multicasting

1. STATUS OF THIS MEMO

This memo specifies the extensions required of a host implementation of the Internet Protocol (IP) to support multicasting. It is the recommended standard for IP multicasting in the Internet. Distribution of this memo is unlimited.

2. INTRODUCTION

IP multicasting is the transmission of an IP datagram to a "host group", a set of zero or more hosts identified by a single IP destination address. A multicast datagram is delivered to all members of its destination host group with the same "best-efforts" reliability as regular unicast IP datagrams, i.e., the datagram is not guaranteed to arrive intact at all members of the destination group or in the same order relative to other datagrams.

The membership of a host group is dynamic; that is, hosts may join and leave groups at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time. A host need not be a member of a group to send datagrams to it.

A host group may be permanent or transient. A permanent group has a well-known, administratively assigned IP address. It is the address, not the membership of the group, that is permanent; at any time a permanent group may have any number of members, even zero. Those IP multicast addresses that are not reserved for permanent groups are available for dynamic assignment to transient groups which exist only as long as they have members.

Internetwork forwarding of IP multicast datagrams is handled by "multicast routers" which may be co-resident with, or separate from, internet gateways. A host transmits an IP multicast datagram as a local network multicast which reaches all immediately-neighboring members of the destination host group. If the datagram has an IP time-to-live greater than 1, the multicast router(s) attached to the local network take responsibility for forwarding it towards all other networks that have members of the destination group. On those other member networks that are reachable within the IP time-to-live, an attached multicast router completes delivery by transmitting the

datagram as a local multicast.

This memo specifies the extensions required of a host IP implementation to support IP multicasting, where a "host" is any internet host or gateway other than those acting as multicast routers. The algorithms and protocols used within and between multicast routers are transparent to hosts and will be specified in separate documents. This memo also does not specify how local network multicasting is accomplished for all types of network, although it does specify the required service interface to an arbitrary local network and gives an Ethernet specification as an example. Specifications for other types of network will be the subject of future memos.

3. LEVELS OF CONFORMANCE

There are three levels of conformance to this specification:

Level 0: no support for IP multicasting.

There is, at this time, no requirement that all IP implementations support IP multicasting. Level 0 hosts will, in general, be unaffected by multicast activity. The only exception arises on some types of local network, where the presence of level 1 or 2 hosts may cause misdelivery of multicast IP datagrams to level 0 hosts. Such datagrams can easily be identified by the presence of a class D IP address in their destination address field; they should be quietly discarded by hosts that do not support IP multicasting. Class D addresses are described in section 4 of this memo.

Level 1: support for sending but not receiving multicast IP datagrams.

Level 1 allows a host to partake of some multicast-based services, such as resource location or status reporting, but it does not allow a host to join any host groups. An IP implementation may be upgraded from level 0 to level 1 very easily and with little new code. Only sections 4, 5, and 6 of this memo are applicable to level 1 implementations.

Level 2: full support for IP multicasting.

Level 2 allows a host to join and leave host groups, as well as send IP datagrams to host groups. It requires implementation of the Internet Group Management Protocol (IGMP) and extension of the IP and local network service interfaces within the host. All of the following sections of this memo are applicable to level 2 implementations.

4. HOST GROUP ADDRESSES

Host groups are identified by class D IP addresses, i.e., those with "1110" as their high-order four bits. Class E IP addresses, i.e., those with "1111" as their high-order four bits, are reserved for future addressing modes.

In Internet standard "dotted decimal" notation, host group addresses range from 224.0.0.0 to 239.255.255.255. The address 224.0.0.0 is guaranteed not to be assigned to any group, and 224.0.0.1 is assigned to the permanent group of all IP hosts (including gateways). This is used to address all multicast hosts on the directly connected network. There is no multicast address (or any other IP address) for all hosts on the total Internet. The addresses of other well-known, permanent groups are to be published in "Assigned Numbers".

Appendix II contains some background discussion of several issues related to host group addresses.

5. MODEL OF A HOST IP IMPLEMENTATION

The multicast extensions to a host IP implementation are specified in terms of the layered model illustrated below. In this model, ICMP and (for level 2 hosts) IGMP are considered to be implemented within the IP module, and the mapping of IP addresses to local network addresses is considered to be the responsibility of local network modules. This model is for expository purposes only, and should not be construed as constraining an actual implementation.

```
         |                                                         |
         |                Upper-Layer Protocol Modules             |
         |_________________________________________________________|

     --------------------- IP Service Interface -----------------------
          _________________________________________________________
         |                             |              |            |
         |                             |     ICMP     |    IGMP    |
         |             IP              |______________|____________|
         |           Module                                        |
         |                                                         |
         |_________________________________________________________|

     ---------------- Local Network Service Interface -----------------
          _________________________________________________________
         |                             |                           |
         |            Local            | IP-to-local address mapping |
         |           Network           |         (e.g., ARP)       |
         |           Modules           |___________________________|
         |       (e.g., Ethernet)                                  |
         |                                                         |
```

To provide level 1 multicasting, a host IP implementation must support the transmission of multicast IP datagrams. To provide level 2 multicasting, a host must also support the reception of multicast IP datagrams. Each of these two new services is described in a separate section, below. For each service, extensions are specified for the IP service interface, the IP module, the local network service interface, and an Ethernet local network module. Extensions to local network modules other than Ethernet are mentioned briefly, but are not specified in detail.

6. SENDING MULTICAST IP DATAGRAMS

6.1. Extensions to the IP Service Interface

Multicast IP datagrams are sent using the same "Send IP" operation used to send unicast IP datagrams; an upper-layer protocol module merely specifies an IP host group address, rather than an individual IP address, as the destination. However, a number of extensions may be necessary or desirable.

First, the service interface should provide a way for the upper-layer protocol to specify the IP time-to-live of an outgoing multicast datagram, if such a capability does not already exist. If the upper-layer protocol chooses not to specify a time-to-live, it should default to 1 for all multicast IP datagrams, so that an explicit choice is required to multicast beyond a single network.

Second, for hosts that may be attached to more than one network, the service interface should provide a way for the upper-layer protocol to identify which network interface is be used for the multicast transmission. Only one interface is used for the initial transmission; multicast routers are responsible for forwarding to any other networks, if necessary. If the upper-layer protocol chooses not to identify an outgoing interface, a default interface should be used, preferably under the control of system management.

Third (level 2 implementations only), for the case in which the host is itself a member of a group to which a datagram is being sent, the service interface should provide a way for the upper-layer protocol to inhibit local delivery of the datagram; by default, a copy of the datagram is looped back. This is a performance optimization for upper-layer protocols that restrict the membership of a group to one process per host (such as a routing protocol), or that handle loopback of group communication at a higher layer (such as a multicast transport protocol).

6.2. Extensions to the IP Module

To support the sending of multicast IP datagrams, the IP module must be extended to recognize IP host group addresses when routing outgoing datagrams. Most IP implementations include the following logic:

```
if IP-destination is on the same local network,
   send datagram locally to IP-destination
else
   send datagram locally to GatewayTo( IP-destination )
```


To allow multicast transmissions, the routing logic must be changed to:

```
if IP-destination is on the same local network
or IP-destination is a host group,
   send datagram locally to IP-destination
else
   send datagram locally to GatewayTo( IP-destination )
```

If the sending host is itself a member of the destination group on the outgoing interface, a copy of the outgoing datagram must be looped-back for local delivery, unless inhibited by the sender. (Level 2 implementations only.)

The IP source address of the outgoing datagram must be one of the individual addresses corresponding to the outgoing interface.

A host group address must never be placed in the source address field or anywhere in a source route or record route option of an outgoing IP datagram.

6.3. Extensions to the Local Network Service Interface

No change to the local network service interface is required to support the sending of multicast IP datagrams. The IP module merely specifies an IP host group destination, rather than an individual IP destination, when it invokes the existing "Send Local" operation.

6.4. Extensions to an Ethernet Local Network Module

The Ethernet directly supports the sending of local multicast packets by allowing multicast addresses in the destination field of Ethernet packets. All that is needed to support the sending of multicast IP datagrams is a procedure for mapping IP host group addresses to Ethernet multicast addresses.

An IP host group address is mapped to an Ethernet multicast address by placing the low-order 23-bits of the IP address into the low-order 23 bits of the Ethernet multicast address 01-00-5E-00-00-00 (hex). Because there are 28 significant bits in an IP host group address, more than one host group address may map to the same Ethernet multicast address.

6.5. Extensions to Local Network Modules other than Ethernet

Other networks that directly support multicasting, such as rings or buses conforming to the IEEE 802.2 standard, may be handled the same

way as Ethernet for the purpose of sending multicast IP datagrams. For a network that supports broadcast but not multicast, such as the Experimental Ethernet, all IP host group addresses may be mapped to a single local broadcast address (at the cost of increased overhead on all local hosts). For a point-to-point link joining two hosts (or a host and a multicast router), multicasts should be transmitted exactly like unicasts. For a store-and-forward network like the ARPANET or a public X.25 network, all IP host group addresses might be mapped to the well-known local address of an IP multicast router; a router on such a network would take responsibility for completing multicast delivery within the network as well as among networks.

7. RECEIVING MULTICAST IP DATAGRAMS

7.1. Extensions to the IP Service Interface

Incoming multicast IP datagrams are received by upper-layer protocol modules using the same "Receive IP" operation as normal, unicast datagrams. Selection of a destination upper-layer protocol is based on the protocol field in the IP header, regardless of the destination IP address. However, before any datagrams destined to a particular group can be received, an upper-layer protocol must ask the IP module to join that group. Thus, the IP service interface must be extended to provide two new operations:

```
JoinHostGroup  ( group-address, interface )

LeaveHostGroup ( group-address, interface )
```

The JoinHostGroup operation requests that this host become a member of the host group identified by "group-address" on the given network interface. The LeaveGroup operation requests that this host give up its membership in the host group identified by "group-address" on the given network interface. The interface argument may be omitted on hosts that support only one interface. For hosts that may be attached to more than one network, the upper-layer protocol may choose to leave the interface unspecified, in which case the request will apply to the default interface for sending multicast datagrams (see section 6.1).

It is permissible to join the same group on more than one interface, in which case duplicate multicast datagrams may be received. It is also permissible for more than one upper-layer protocol to request membership in the same group.

Both operations should return immediately (i.e., they are non-blocking operations), indicating success or failure. Either operation may fail due to an invalid group address or interface

identifier. JoinHostGroup may fail due to lack of local resources. LeaveHostGroup may fail because the host does not belong to the given group on the given interface. LeaveHostGroup may succeed, but the membership persist, if more than one upper-layer protocol has requested membership in the same group.

7.2. Extensions to the IP Module

To support the reception of multicast IP datagrams, the IP module must be extended to maintain a list of host group memberships associated with each network interface. An incoming datagram destined to one of those groups is processed exactly the same way as datagrams destined to one of the host's individual addresses.

Incoming datagrams destined to groups to which the host does not belong are discarded without generating any error report or log entry. On hosts with more than one network interface, if a datagram arrives via one interface, destined for a group to which the host belongs only on a different interface, the datagram is quietly discarded. (These cases should occur only as a result of inadequate multicast address filtering in a local network module.)

An incoming datagram is not rejected for having an IP time-to-live of 1 (i.e., the time-to-live should not automatically be decremented on arriving datagrams that are not being forwarded). An incoming datagram with an IP host group address in its source address field is quietly discarded. An ICMP error message (Destination Unreachable, Time Exceeded, Parameter Problem, Source Quench, or Redirect) is never generated in response to a datagram destined to an IP host group.

The list of host group memberships is updated in response to JoinHostGroup and LeaveHostGroup requests from upper-layer protocols. Each membership should have an associated reference count or similar mechanism to handle multiple requests to join and leave the same group. On the first request to join and the last request to leave a group on a given interface, the local network module for that interface is notified, so that it may update its multicast reception filter (see section 7.3).

The IP module must also be extended to implement the IGMP protocol, specified in Appendix I. IGMP is used to keep neighboring multicast routers informed of the host group memberships present on a particular local network. To support IGMP, every level 2 host must join the "all-hosts" group (address 224.0.0.1) on each network interface at initialization time and must remain a member for as long as the host is active.

(Datagrams addressed to the all-hosts group are recognized as a special case by the multicast routers and are never forwarded beyond a single network, regardless of their time-to-live. Thus, the all-hosts address may not be used as an internet-wide broadcast address. For the purpose of IGMP, membership in the all-hosts group is really necessary only while the host belongs to at least one other group. However, it is specified that the host shall remain a member of the all-hosts group at all times because (1) it is simpler, (2) the frequency of reception of unnecessary IGMP queries should be low enough that overhead is negligible, and (3) the all-hosts address may serve other routing-oriented purposes, such as advertising the presence of gateways or resolving local addresses.)

7.3. Extensions to the Local Network Service Interface

Incoming local network multicast packets are delivered to the IP module using the same "Receive Local" operation as local network unicast packets. To allow the IP module to tell the local network module which multicast packets to accept, the local network service interface is extended to provide two new operations:

```
JoinLocalGroup  ( group-address )

LeaveLocalGroup ( group-address )
```

where "group-address" is an IP host group address. The JoinLocalGroup operation requests the local network module to accept and deliver up subsequently arriving packets destined to the given IP host group address. The LeaveLocalGroup operation requests the local network module to stop delivering up packets destined to the given IP host group address. The local network module is expected to map the IP host group addresses to local network addresses as required to update its multicast reception filter. Any local network module is free to ignore LeaveLocalGroup requests, and may deliver up packets destined to more addresses than just those specified in JoinLocalGroup requests, if it is unable to filter incoming packets adequately.

The local network module must not deliver up any multicast packets that were transmitted from that module; loopback of multicasts is handled at the IP layer or higher.

7.4. Extensions to an Ethernet Local Network Module

To support the reception of multicast IP datagrams, an Ethernet module must be able to receive packets addressed to the Ethernet multicast addresses that correspond to the host's IP host group addresses. It is highly desirable to take advantage of any address

filtering capabilities that the Ethernet hardware interface may have, so that the host receives only those packets that are destined to it.

Unfortunately, many current Ethernet interfaces have a small limit on the number of addresses that the hardware can be configured to recognize. Nevertheless, an implementation must be capable of listening on an arbitrary number of Ethernet multicast addresses, which may mean "opening up" the address filter to accept all multicast packets during those periods when the number of addresses exceeds the limit of the filter.

For interfaces with inadequate hardware address filtering, it may be desirable (for performance reasons) to perform Ethernet address filtering within the software of the Ethernet module. This is not mandatory, however, because the IP module performs its own filtering based on IP destination addresses.

7.5. Extensions to Local Network Modules other than Ethernet

Other multicast networks, such as IEEE 802.2 networks, can be handled the same way as Ethernet for the purpose of receiving multicast IP datagrams. For pure broadcast networks, such as the Experimental Ethernet, all incoming broadcast packets can be accepted and passed to the IP module for IP-level filtering. On point-to-point or store-and-forward networks, multicast IP datagrams will arrive as local network unicasts, so no change to the local network module should be necessary.

APPENDIX I. INTERNET GROUP MANAGEMENT PROTOCOL (IGMP)

The Internet Group Management Protocol (IGMP) is used by IP hosts to report their host group memberships to any immediately-neighboring multicast routers. IGMP is an asymmetric protocol and is specified here from the point of view of a host, rather than a multicast router. (IGMP may also be used, symmetrically or asymmetrically, between multicast routers. Such use is not specified here.)

Like ICMP, IGMP is a integral part of IP. It is required to be implemented by all hosts conforming to level 2 of the IP multicasting specification. IGMP messages are encapsulated in IP datagrams, with an IP protocol number of 2. All IGMP messages of concern to hosts have the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Type  |    Unused     |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Group Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version

This memo specifies version 1 of IGMP. Version 0 is specified in RFC-988 and is now obsolete.

Type

There are two types of IGMP message of concern to hosts:

1 = Host Membership Query
2 = Host Membership Report

Unused

Unused field, zeroed when sent, ignored when received.

Checksum

The checksum is the 16-bit one's complement of the one's complement sum of the 8-octet IGMP message. For computing the checksum, the checksum field is zeroed.

Group Address

In a Host Membership Query message, the group address field

is zeroed when sent, ignored when received.

In a Host Membership Report message, the group address field holds the IP host group address of the group being reported.

Informal Protocol Description

Multicast routers send Host Membership Query messages (hereinafter called Queries) to discover which host groups have members on their attached local networks. Queries are addressed to the all-hosts group (address 224.0.0.1), and carry an IP time-to-live of 1.

Hosts respond to a Query by generating Host Membership Reports (hereinafter called Reports), reporting each host group to which they belong on the network interface from which the Query was received. In order to avoid an "implosion" of concurrent Reports and to reduce the total number of Reports transmitted, two techniques are used:

1. When a host receives a Query, rather than sending Reports immediately, it starts a report delay timer for each of its group memberships on the network interface of the incoming Query. Each timer is set to a different, randomly-chosen value between zero and D seconds. When a timer expires, a Report is generated for the corresponding host group. Thus, Reports are spread out over a D second interval instead of all occurring at once.

2. A Report is sent with an IP destination address equal to the host group address being reported, and with an IP time-to-live of 1, so that other members of the same group on the same network can overhear the Report. If a host hears a Report for a group to which it belongs on that network, the host stops its own timer for that group and does not generate a Report for that group. Thus, in the normal case, only one Report will be generated for each group present on the network, by the member host whose delay timer expires first. Note that the multicast routers receive all IP multicast datagrams, and therefore need not be addressed explicitly. Further note that the routers need not know which hosts belong to a group, only that at least one host belongs to a group on a particular network.

There are two exceptions to the behavior described above. First, if a report delay timer is already running for a group membership when a Query is received, that timer is not reset to a new random value, but rather allowed to continue running with its current value. Second, a report delay timer is never set for a host's membership in the all-hosts group (224.0.0.1), and that membership is never reported.

If a host uses a pseudo-random number generator to compute the reporting delays, one of the host's own individual IP address should be used as part of the seed for the generator, to reduce the chance of multiple hosts generating the same sequence of delays.

A host should confirm that a received Report has the same IP host group address in its IP destination field and its IGMP group address field, to ensure that the host's own Report is not cancelled by an erroneous received Report. A host should quietly discard any IGMP message of type other than Host Membership Query or Host Membership Report.

Multicast routers send Queries periodically to refresh their knowledge of memberships present on a particular network. If no Reports are received for a particular group after some number of Queries, the routers assume that that group has no local members and that they need not forward remotely-originated multicasts for that group onto the local network. Queries are normally sent infrequently (no more than once a minute) so as to keep the IGMP overhead on hosts and networks very low. However, when a multicast router starts up, it may issue several closely-spaced Queries in order to build up its knowledge of local memberships quickly.

When a host joins a new group, it should immediately transmit a Report for that group, rather than waiting for a Query, in case it is the first member of that group on the network. To cover the possibility of the initial Report being lost or damaged, it is recommended that it be repeated once or twice after short delays. (A simple way to accomplish this is to act as if a Query had been received for that group only, setting the group's random report delay timer. The state transition diagram below illustrates this approach.)

Note that, on a network with no multicast routers present, the only IGMP traffic is the one or more Reports sent whenever a host joins a new group.

State Transition Diagram

IGMP behavior is more formally specified by the state transition diagram below. A host may be in one of three possible states, with respect to any single IP host group on any single network interface:

- Non-Member state, when the host does not belong to the group on the interface. This is the initial state for all memberships on all network interfaces; it requires no storage in the host.

- Delaying Member state, when the host belongs to the group on the interface and has a report delay timer running for that membership.

- Idle Member state, when the host belongs to the group on the interface and does not have a report delay timer running for that membership.

There are five significant events that can cause IGMP state transitions:

- "join group" occurs when the host decides to join the group on the interface. It may occur only in the Non-Member state.

- "leave group" occurs when the host decides to leave the group on the interface. It may occur only in the Delaying Member and Idle Member states.

- "query received" occurs when the host receives a valid IGMP Host Membership Query message. To be valid, the Query message

      must be at least 8 octets long, have a correct IGMP
      checksum and have an IP destination address of 224.0.0.1.
      A single Query applies to all memberships on the
      interface from which the Query is received. It is ignored for
      memberships in the Non-Member or Delaying Member state.

    - "report received" occurs when the host receives a valid IGMP
      Host Membership Report message. To be valid, the Report
      message must be at least 8 octets long, have a correct IGMP
      checksum, and contain the same IP host group address in its IP
      destination field and its IGMP group address field. A Report
      applies only to the membership in the group identified by the
      Report, on the interface from which the Report is received.
      It is ignored for memberships in the Non-Member or Idle Member
      state.

    - "timer expired" occurs when the report delay timer for the
      group on the interface expires. It may occur only in the
      Delaying Member state.

All other events, such as receiving invalid IGMP messages, or IGMP messages other than Query or Report, are ignored in all states.

There are three possible actions that may be taken in response to the above events:

- "send report" for the group on the interface.

- "start timer" for the group on the interface, using a random delay value between 0 and D seconds.

- "stop timer" for the group on the interface.

In the following diagram, each state transition arc is labelled with the event that causes the transition, and, in parentheses, any actions taken during the transition.

```
                           ________________
                          |                |
                          |                |
                          |                |
                          |                |
               --------->|   Non-Member    |<---------
              |           |                |          |
              |           |                |          |
              |           |                |          |
              |           |________________|          |
              |                   |                   |
              | leave group       | join group        | leave group
```

```
                  | (stop timer)      |(send report,     |
                  |                   | start timer)     |
          ________|________           |          ________|________
         |                 |<---------          |                 |
         |                 |                    |                 |
         |                 |<-------------------|                 |
         |                 |  query received    |                 |
         | Delaying Member |   (start timer)    |   Idle Member   |
         |                 |------------------->|                 |
         |                 |  report received   |                 |
         |                 |   (stop timer)     |                 |
         |_________________|------------------->|_________________|
                               timer expired
                               (send report)
```

The all-hosts group (address 224.0.0.1) is handled as a special case. The host starts in Idle Member state for that group on every interface, never transitions to another state, and never sends a report for that group.

Protocol Parameters

The maximum report delay, D, is 10 seconds.

APPENDIX II. HOST GROUP ADDRESS ISSUES

This appendix is not part of the IP multicasting specification, but provides background discussion of several issues related to IP host group addresses.

Group Address Binding

The binding of IP host group addresses to physical hosts may be considered a generalization of the binding of IP unicast addresses. An IP unicast address is statically bound to a single local network interface on a single IP network. An IP host group address is dynamically bound to a set of local network interfaces on a set of IP networks.

It is important to understand that an IP host group address is NOT bound to a set of IP unicast addresses. The multicast routers do not need to maintain a list of individual members of each host group. For example, a multicast router attached to an Ethernet need associate only a single Ethernet multicast address with each host group having local members, rather than a list of the members' individual IP or Ethernet addresses.

Allocation of Transient Host Group Addresses

This memo does not specify how transient group address are allocated. It is anticipated that different portions of the IP transient host group address space will be allocated using different techniques. For example, there may be a number of servers that can be contacted to acquire a new transient group address. Some higher-level protocols (such as VMTP, specified in RFC-1045) may generate higher-level transient "process group" or "entity group" addresses which are then algorithmically mapped to a subset of the IP transient host group addresses, similarly to the way that IP host group addresses are mapped to Ethernet multicast addresses. A portion of the IP group address space may be set aside for random allocation by applications that can tolerate occasional collisions with other multicast users, perhaps generating new addresses until a suitably "quiet" one is found.

In general, a host cannot assume that datagrams sent to any host group address will reach only the intended hosts, or that datagrams received as a member of a transient host group are intended for the recipient. Misdelivery must be detected at a level above IP, using higher-level identifiers or authentication tokens. Information transmitted to a host group address should be encrypted or governed by administrative routing controls if the sender is concerned about unwanted listeners.

Author's Address

Steve Deering
Stanford University
Computer Science Department
Stanford, CA 94305-2140

Phone: (415) 723-9427

EMail: deering@PESCADERO.STANFORD.EDU