            Ipsilon Flow Management Protocol Specification for IPv4
                              Version 1.0

Status of this Memo

IESG Note:

   This memo documents a private protocol for IPv4-based flows.  This
   protocol is NOT the product of an IETF working group nor is it a
   standards track document.  It has not necessarily benefited from the
   widespread and in depth community review that standards track
   documents receive.

Abstract

   The Ipsilon Flow Management Protocol (IFMP), is a protocol for
   allowing a node to instruct an adjacent node to attach a layer 2
   label to a specified IP flow.  The label allows more efficient access
   to cached routing information for that flow.  The label can also
   enable a node to switch further packets belonging to the specified
   flow at layer 2 rather than forwarding them at layer 3.

Table of Contents

1. Introduction

    The Ipsilon Flow Management Protocol (IFMP), is a protocol for
    instructing an adjacent node to attach a layer 2 label to a specified
    IP flow. The label allows more efficient access to cached routing
    information for that flow and it allows the flow to be switched
    rather than routed in certain cases.

    If a network node's upstream and downstream links both redirect a
    flow at the node, then the node can switch the flow at the data link
    layer rather than forwarding it at the network layer.  The label
    space is managed at the downstream end of each link and redirection
    messages are sent upstream to associate a particular flow with a
    given label.  Each direction of transmission on a link is treated
    separately.

    If the flow is not refreshed by the time the lifetime field in the
    redirect message expires, then the association between the flow and
    the label is discarded.  A flow is refreshed by sending a redirect
    message, identical to the original, before the lifetime expires.

    Several flow types may be specified.  Each flow type specifies the
    set of fields from the packet header that are used to identify a
    flow.  There must be an ordering amongst the different flow types
    such that a most specific match operation may be performed.

    A particular flow is specified by a flow identifier.  The flow
    identifier for that flow gives the contents of the set of fields from
    the packet header as defined for the flow type to which it belongs.

    This document specifies the IFMP protocol for IPv4 on a point-to-
    point link.  The definition of labels, and the encapsulation of
    flows, are specified in a separate document for each specific data
    link technology.  The specification for ATM data links is given in
    [ENCAP].

2. Flow Types

    A flow is a sequence of packets that are sent from a particular
    source to a particular (unicast or multicast) destination and that
    are related in terms of their routing and any logical handling policy
    they may require.

A flow is identified by its flow identifier.

Several different flow types can be defined.  The particular set of
fields from the packet header used to identify a flow constitutes the
flow type.  The values of these fields, for a particular flow,
constitutes the flow identifier for that flow.  The values of these
fields must be invariant in all packets belonging to the same flow at
any point in the network.

Flow types are sub- or super-sets of each other such that there is a
clear hierarchy of flow types.  This permits a most specific match
operation to be performed.  (If additional flow types are defined in
the future that are not fully ordered then the required behavior will
be defined.) Each flow type also specifies an encapsulation that is
to be used after a flow of this type is redirected.  The
encapsulations for each flow type are specified in a separate
document for each specific data link technology.  The encapsulations
for flows over ATM data links are given in [ENCAP].

Three flow types are defined in this version of the protocol:

Flow Type 0

   Flow Type 0 is used to change the encapsulation of IPv4 packets
   from the default encapsulation.

   For Flow Type 0: Flow Type = 0 and Flow ID Length = 0.
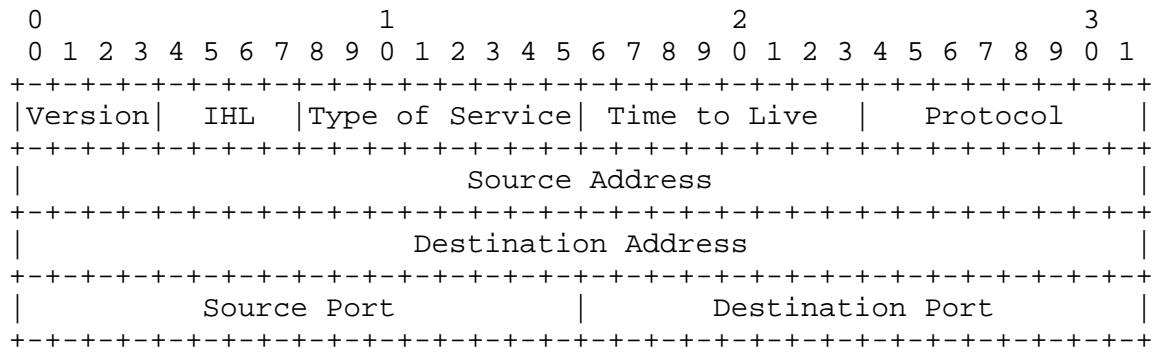
   The Flow Identifier for Flow Type 0 is null (zero length).

Flow Type 1

   Flow Type 1 is designed for protocols such as UDP and TCP in which
   the first four octets after the IPv4 header specify a Source Port
   number and a Destination Port number.

   For Flow Type 1, Flow Type = 1 and Flow ID Length = 4 (32 bit
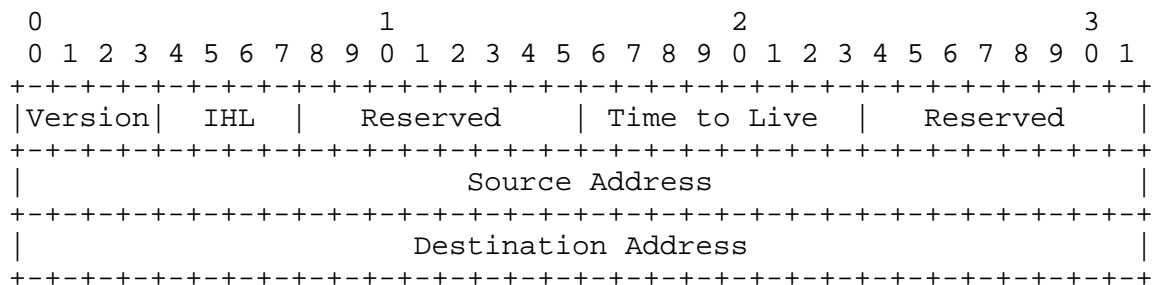   words).

   The format of the Flow Identifier for Flow Type 1 is:

```
 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| Time to Live  |    Protocol   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Destination Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Flow Type 2

   For Flow Type 2, Flow Type = 2 and Flow ID Length = 3 (32 bit
   words).

   The format of the Flow Identifier for Flow Type 2 is:

```
 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  |   Reserved    | Time to Live  |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Destination Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   The Reserved fields are unused and should be set to zero by the
   sender and ignored by the receiver.

3. IFMP Adjacency Protocol

   The IFMP Adjacency Protocol allows a host or router to discover the
   identity of a peer at the other end of a link.  It is also used to
   synchronize state across the link, to detect when the peer at the
   other end of the link changes, and to exchange a list of IP addresses
   assigned to the link.

3.1 Packet Format

   All IFMP messages belonging to the Adjacency Protocol must be
   encapsulated within an IPv4 packet and must be sent to the IP limited
   broadcast address (255.255.255.255).  The Protocol field in the IP
   header must contain the value 101 (decimal) indicating that the IP
   packet contains an IFMP message.  The Time to Live (TTL) field in the
   IP header must be set to 1.

All IFMP messages belonging to the adjacency protocol have the
following structure:

```
 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   |    Op Code    |            Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sender Instance                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Peer Instance                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Peer Identity                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Peer Next Sequence Number                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Reserved             |    Reserved   | Max Ack Intvl |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                         Address List                          ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
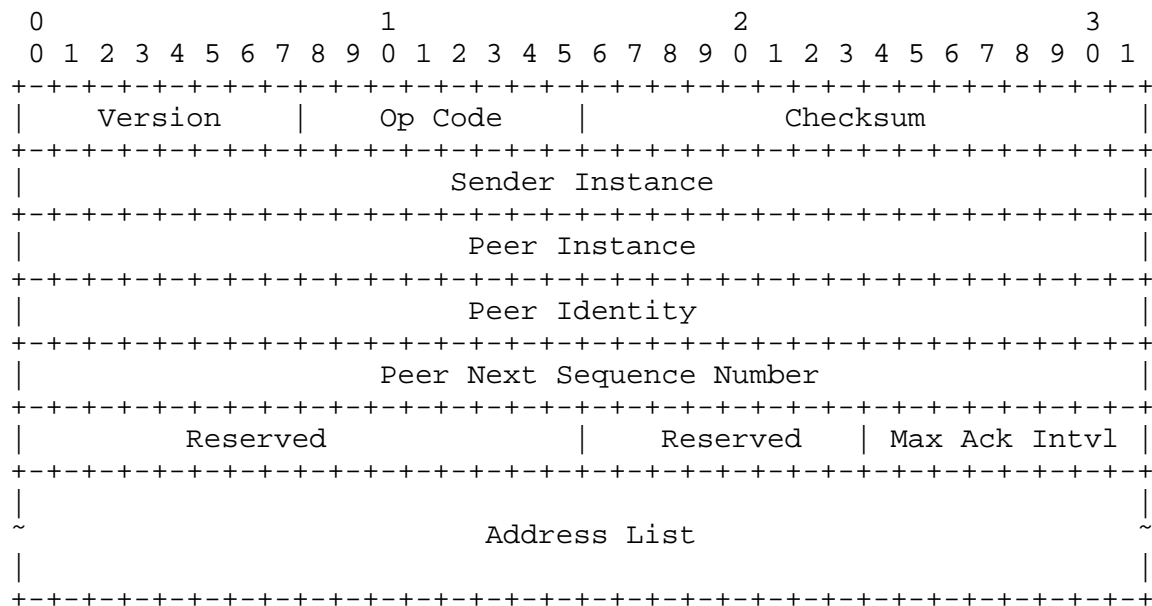
Version

        The IFMP protocol version number.  The current Version = 1.

Op Code

        Specifies the function of the message.  Four Op Codes are
        defined for the IFMP Adjacency Protocol:

            SYN:    Op Code = 0
            SYNACK: Op Code = 1
            RSTACK: Op Code = 2
            ACK:    Op Code = 3

Checksum

        The 16-bit one's complement of the one's complement sum of
        a pseudo header of information from the IP header and the
        IFMP message itself.  The pseudo header, conceptually
        prefixed to the IFMP message, contains the Source Address,
        the Destination Address, and the Protocol fields from the
        IPv4 header, and the total length of the IFMP message
        starting with the Version field (this is equivalent to the
        value of the Total Length field from the IPv4 header minus
        the length of the IPv4 header itself).

Sender Instance
        For the SYN, SYNACK, and ACK messages, is the sender's
        instance number for the link.  The receiver uses this to
        detect when the link comes back up after going down or when
        the identity of the peer at the other end of the link
        changes.  The instance number is a 32 bit number that is
        guaranteed to be unique within the recent past and to
        change when the link or node comes back up after going
        down.  It is used in a similar manner to the initial
        sequence number (ISN) in TCP [RFC 793].  Zero is not a
        valid instance number.  For the RSTACK message the Sender
        Instance field is set to the value of the Peer Instance
        field from the incoming message that caused an RSTACK
        message to be generated.

Peer Instance
        For the SYN, SYNACK, and ACK messages, is what the sender
        believes is the peer's current instance number for the
        link.  If the sender of the message does not know the
        peer's current instance number for the link, the sender
        must set this field to zero.  For the RSTACK message the
        Peer Instance field is set to the value of the Sender
        Instance field from the incoming message that caused an
        RSTACK message to be generated.

Peer Identity
        For the SYN, SYNACK, and ACK messages, is the IP address of
        the peer that the sender of the message believes is at the
        other end of the link.  The Peer Identity is taken from the
        Source IP Address of the IP header of a SYN or a SYNACK
        message.  If the sender of the message does not know the IP
        address of the peer at the other end of the link, the
        sender must set set this field to zero.  For the RSTACK
        message, the Peer Identity field is set to the value of the
        Source Address field from the IP header of the incoming
        message that caused an RSTACK message to be generated.

Peer Next Sequence Number
        Gives the value of the peer's Sequence Number that the
        sender of the IFMP Adjacency Protocol message expects to
        arrive in the next IFMP Redirection Protocol message.  If a
        node is in the ESTAB state, and the value of the Peer Next
        Sequence Number in an incoming ACK message is greater than
        the value of the Sequence Number plus one, from the last
        IFMP Redirection Protocol message transmitted out of the
        port on which the incoming ACK message was received, the
        link should be reset.  The procedure to reset the link is
        defined in section 3.2.

Max Ack Intvl
        Maximum Acknowledgement Interval is the maximum amount of
        time the sender of the message will wait until transmitting
        an ACK message.

Address List
        A list of one or more IP addresses that are assigned to the
        link by the sender of the message.  The list must have at
        least one entry that is identical to the Source Address in
        the IP header.  The contents of this list are not used by
        the IFMP protocol but can be made available to the routing
        protocol.

## 3.2 Procedure

The IFMP Adjacency Protocol is described by the rules and state
tables given in this section.

The rules and state tables use the following operations:

o The "Update Peer Verifier" operation is defined as storing the
  Sender Instance and the Source IP Address from a SYN or SYNACK
  message received from the peer on a particular port.

o The procedure "Reset the link" is defined as:

        1. Generate a new instance number for the link
        2. Delete the peer verifier (set the stored values of Sender
           Instance and Source IP Address of the peer to zero)
        3. Set Sequence Number and Peer Next Sequence Number to zero
        4. Send a SYN message
        5. Enter the SYNSENT state

o The state tables use the following Boolean terms and operators:

    A    The Sender Instance in the incoming message matches the
         value stored from a previous message by the "Update Peer
         Verifier" operation for the port on which the incoming
         message is received.

    B    The Sender Instance and the Source IP Address in the
         incoming message matches the value stored from a previous
         message by the "Update Peer Verifier" operation for the
         port on which the incoming message is received.

         C      The Peer Instance and Peer Identity in the incoming message
                matches the value of the Sender Instance and the Source IP
                Address currently in use for all SYN, SYNACK, and ACK
                messages transmitted out of the port on which the incoming
                message was received.

         "&&" Represents the logical AND operation

         "||" Represents the logical OR operation

         "!" Represents the logical negation (NOT) operation.

  o A timer is required for the periodic generation of SYN, SYNACK,
    and ACK messages.  The period of the timer is unspecified but a
    value of one second is suggested.

    There are two independent events: the timer expires, and a packet
    arrives.  The processing rules for these events are:

       Timer Expires:    Reset Timer
                         If state = SYNSENT Send SYN
                         If state = SYNRCVD Send SYNACK
                         If state = ESTAB   Send ACK

       Packet Arrives:   If incoming message is an RSTACK
                             If A && C && !SYNSENT
                                Reset the link
                             Else Discard the message
                         Else the following State Tables.


  o State synchronization across a link is considered to be achieved
    when a node reaches the ESTAB state.

State Tables

   State: SYNSENT

```
+==============================================================================+
|     Condition      |                Action                | New State |
+====================+======================================+===========+
|   SYNACK && C      | Update Peer Verifier; Send ACK       |   ESTAB   |
+--------------------+--------------------------------------+-----------+
|   SYNACK && !C     |             Send RSTACK              |  SYNSENT  |
+--------------------+--------------------------------------+-----------+
|       SYN          | Update Peer Verifier; Send SYNACK    |  SYNRCVD  |
+--------------------+--------------------------------------+-----------+
|       ACK          |             Send RSTACK              |  SYNSENT  |
+==============================================================================+
```

   State: SYNRCVD

```
+==============================================================================+
|     Condition      |                Action                | New State |
+====================+======================================+===========+
|   SYNACK && C      | Update Peer Verifier; Send ACK       |   ESTAB   |
+--------------------+--------------------------------------+-----------+
|   SYNACK && !C     |             Send RSTACK              |  SYNRCVD  |
+--------------------+--------------------------------------+-----------+
|       SYN          | Update Peer Verifier; Send SYNACK    |  SYNRCVD  |
+--------------------+--------------------------------------+-----------+
|   ACK && B && C    |             Send ACK                |   ESTAB   |
+--------------------+--------------------------------------+-----------+
|  ACK && !(B && C)  |             Send RSTACK              |  SYNRCVD  |
+==============================================================================+
```

   State: ESTAB

```
+==============================================================================+
|     Condition      |                Action                | New State |
+====================+======================================+===========+
|  SYN || SYNACK     |           Send ACK (note 1)          |   ESTAB   |
+--------------------+--------------------------------------+-----------+
|  ACK && B && C     |           Send ACK (note 1)          |   ESTAB   |
+--------------------+--------------------------------------+-----------+
|  ACK && !(B && C)  |             Send RSTACK              |   ESTAB   |
+==============================================================================+
```

 Note 1: No more than one ACK should be sent within any time period of
         length defined by the timer.

4. IFMP Redirection Protocol

   A sender encapsulates within an IPv4 packet all IFMP messages
   belonging to the Redirection Protocol.  The sender sends these
   messages to the unicast IP address of the peer at the other end of
   the link. The IP address of the peer is obtained from the adjacency
   protocol.  The Protocol field in the IP header must contain the value
   101 (decimal) indicating that the IP packet contains an IFMP message.
   The Time to Live (TTL) field in the IP header must be set to 1.

   All IFMP Redirection Protocol messages have the following structure:

```
 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    |    Op Code    |            Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sender Instance                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Peer Instance                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                         Message Body                          ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
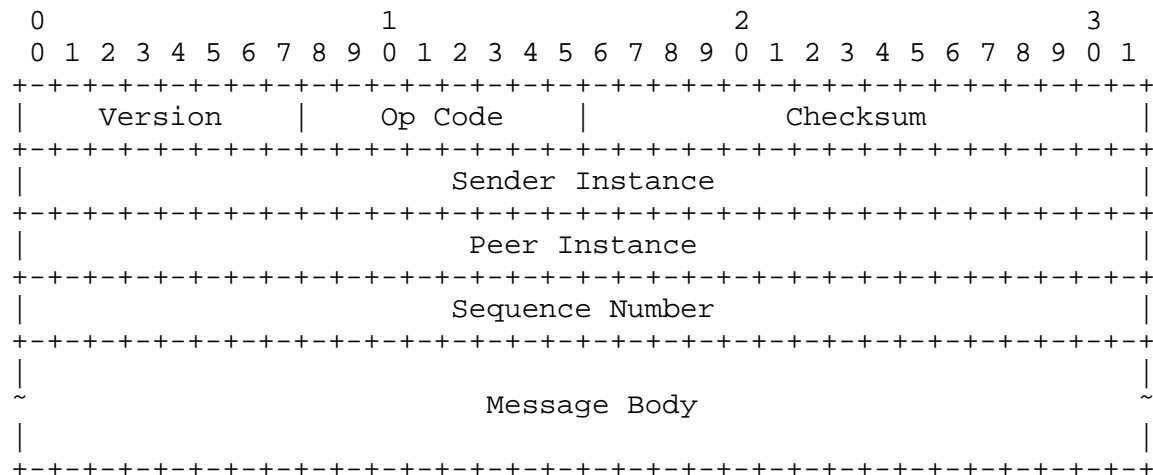
   Version
           The IFMP protocol version number, currently Version = 1.

   Op Code
           This field gives the message type.  Five message types are
           currently defined for the IFMP Redirection Protocol:

               REDIRECT:      Op Code = 4
               RECLAIM:       Op Code = 5
               RECLAIM ACK:   Op Code = 6
               LABEL RANGE:   Op Code = 7
               ERROR:         Op Code = 8

   Checksum
           The 16-bit one's complement of the one's complement sum of
           a pseudo header of information from the IP header, and the
           IFMP message itself.  The pseudo header, conceptually
           prefixed to the IFMP message, contains the Source Address,
           the Destination Address, and the Protocol fields from the

IPv4 header, and the total length of the IFMP message
starting with the version field (this is equivalent to the
value of the Total Length field from the IPv4 header minus
the length of the IPv4 header itself).

Sender Instance
        The sender's instance number for the link from the IFMP
        Adjacency Protocol.

Peer Instance
        What the sender believes is the peer's current instance
        number for the link from the IFMP Adjacency protocol.

Sequence Number
        The sender must increment by one, modulo 2**32, for every
        IFMP Redirection Protocol message sent across a link.  It
        allows the receiver to process IFMP Redirection Protocol
        messages in order.  The Sequence Number is set to zero when
        a node resets the link.

Message Body
        Contains a list of one or more IFMP Redirection Protocol
        message elements.  All of the message elements in the list
        have the same message type because the Op Code field
        applies to the entire IFMP message.  The number of message
        elements included in a single packet must not cause the
        total size of the IFMP message to exceed the MTU size of
        the underlying data link.  Only a single message element is
        permitted in a Label Range message or in an Error message.

No IFMP Redirection Protocol messages can be sent across a link until
the IFMP Adjacency Protocol has achieved state synchronization across
that link.  All IFMP Redirection Protocol messages received on a link
that does not currently have state synchronization must be discarded.
For every received IFMP Redirection Protocol message the receiver
must check the Source IP Address from the IP header, the Sender
Instance, and the Peer Instance.  The incoming message must be
discarded if the Sender Instance and the Source IP Address fields do
not match the values stored by the "Update Peer Verifier" operation
of the IFMP Adjacency Protocol for the port on which the message is
received.  The incoming message must also be discarded if the Peer
Instance field does not match the current value for the Sender
Instance of the IFMP Adjacency Protocol.

4.1 Redirect Message

   The Redirect Message element is used to instruct an adjacent node to
   attach one or more given labels to packets belonging to one or more
   specified flows each for a specified period of time.  The Redirect
   message is not acknowledged.

   Each Redirect message element has the following structure:
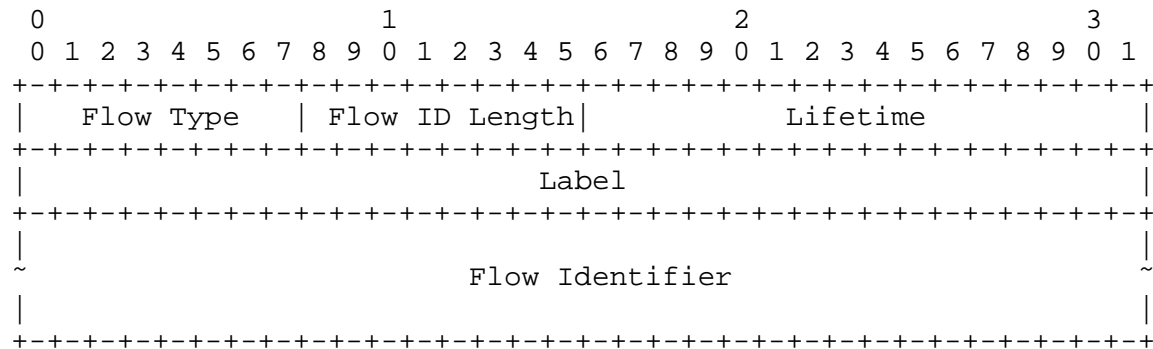
```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Flow Type   | Flow ID Length|            Lifetime           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             Label                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                        Flow Identifier                        ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Flow Type
           Specifies the Flow Type of the flow identifier contained in
           the Flow Identifier field.

   Flow ID Length
           Specifies the length of the Flow Identifier field in
           integer multiples of 32 bit words.

   Lifetime field
           Specifies the length of time, in seconds, for which this
           redirection is valid.  The association of flow identifier
           and label should be discarded at a time no greater than
           that specified by the Lifetime field.  A value of zero is
           not valid.

   Label field
           Contains a 32 bit label.  The format of the label is
           dependent upon the type of physical link across which the
           Redirect message is sent.  (The format of the label for ATM
           data links is specified in [ENCAP].)

   Flow Identifier
           Identifies the flow with which the specified label should
           be associated.  The length of the Flow Identifier field
           must be an integer multiple of 32 bit words to preserve 32
           bit alignment.

A node can send an IFMP message containing one or more Redirect
message elements across a link to its upstream neighbor.  Each
Redirect message element requests that the upstream neighbor
associate a given link-level label to packets belonging to a
specified flow for up to a specified period of time.  A node
receiving an IFMP message that contains one or more Redirect message
elements from an adjacent downstream neighbor can choose to ignore
any or all of the Redirect message elements.  Neither the IFMP
message nor any of the Redirect message elements are acknowledged.
If the node chooses to accept a particular Redirect message element
and to redirect the specified flow, it should attach the label
specified in the Redirect message element to all further packets sent
on that flow until it chooses to do so no longer, or until the
specified lifetime expires.  While the flow remains redirected, the
encapsulation specified by the definition of the Flow Type given in
the Redirect message element must be used for all packets belonging
to that flow.  If the label in a Redirect message element is outside
the range that can be handled across the relevant link, a Label Range
message can be returned to the sender.  The Label Range message
informs the sender of the Redirect message of the range of labels
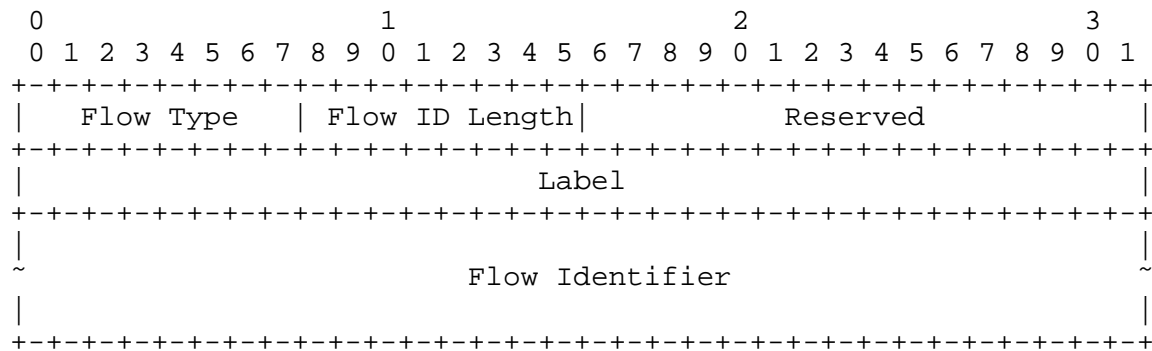that can be sent across the link.

If a Redirect message element is received specifying a flow that is
already redirected, the Label field in the received Redirect message
element must be checked against the label stored for the redirected
flow.  If they agree, the lifetime of the redirected flow is reset to
that contained in the Redirect message element.  If they disagree,
the Redirect message element is ignored, and the flow returned to the
default state.  There is a minimum time between Redirect message
elements specifying the same flow.  The default value is one second.

If a receiving node detects an error in any of the fields of a
Redirect message element, the node must discard that message element
without affecting any other Redirect message elements in the same
IFMP message.  The receiver should return an error message to the
sender only in the case that the receiver does not understand the
version of the IFMP protocol in the received IFMP message or does not
understand a Flow Type in any of the Redirect message elements.  An
Error Message should be returned for each Flow Type that is not
understood.

4.2 Reclaim Message

The Reclaim message element is used by a node to instruct an adjacent
upstream node to unbind one or more flows from the labels to which
they are currently bound, and to release the labels.

Each Reclaim message element has the following structure:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Flow Type   | Flow ID Length|            Reserved           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Label                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                         Flow Identifier                       ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Flow Type
        Specifies the Flow Type of the Flow Identifier contained in
        the Flow ID field.

Flow ID Length
        Specifies the length of the Flow Identifier field in
        integer multiples of 32 bit words.

Reserved
        Field is unused and should be set to zero by the sender and
        ignored by the receiver.

Label
        Field contains the label to be released.

Flow Identifier
        Field contains the flow identifier to be unbound.

A node can send a Reclaim message element to instruct an adjacent
upstream node to unbind a flow from the label to which it is
currently bound, return the flow to the default forwarding state, and
release the label.  Each Reclaim message element applies to a single
flow and a single label.  When the receiver has completed the
operation, it must issue a Reclaim Ack message element.  Reclaim Ack
message elements can be grouped together, in any order, into one or
more IFMP Reclaim Ack messages and returned to the sender as an
acknowledgment that the operation is complete.

If a Reclaim message element is received indicating an unknown flow,
a Reclaim Ack message element must be returned containing the same
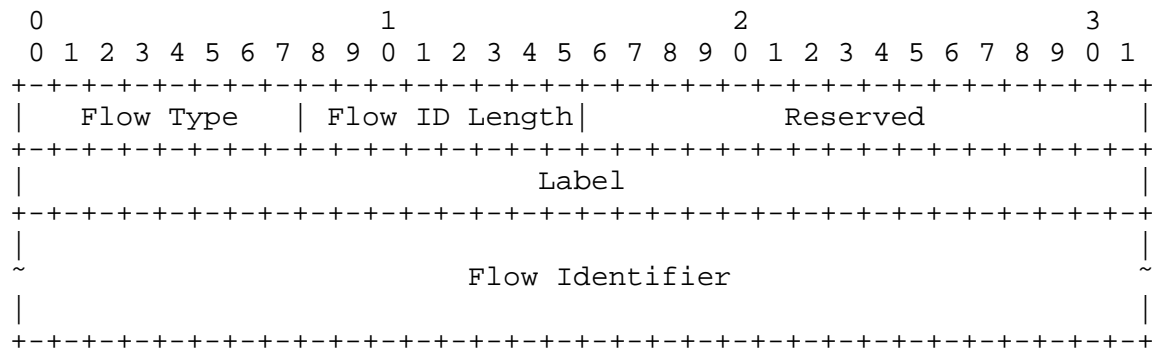Label and Flow Identifier fields from the Reclaim message.

If a Reclaim message element is received indicating a known flow, but
with a Label that is not currently bound to that flow, the flow must
be unbound and returned to the default forwarding state, and a
Reclaim Ack message sent containing the actual label to which the
flow was previously bound.

If the receiver detects an error in any of the fields of a Reclaim
message element, the receiver must discard that message element,
without affecting any other Reclaim message elements in the same
message.  The receiver must return an error message to the sender
only in the case that the receiver does not understand the version of
the IFMP protocol in the received message or does not understand a
Flow Type in one of the Reclaim message elements.

4.3 Reclaim Ack Message

   The Reclaim Ack message element is used by a receiving node to
   acknowledge the successful release of one or more reclaimed labels.

   Each Reclaim Ack message element has the following structure:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Flow Type   | Flow ID Length|             Reserved          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Label                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                        Flow Identifier                        ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Flow Type
           Specifies the Flow Type of the Flow Identifier contained in
           the Flow Identifier field.

   Flow ID Length
           Specifies the length of the Flow Identifier field in
           integer multiples of 32 bit words.

   Reserved
           Field is unused and should be set to zero by the sender and
           ignored by the receiver.

   Label
           Field contains the label released from the flow specified
           by the Flow Identifier.

Flow Identifier
        Field contains the Flow Identifier from the Reclaim message
        element that requested the release of the label specified
        in the Label field.

A Reclaim Ack message element must be sent in response to each
Reclaim message element received.  It is sent to indicate that the
requested flow is now unbound and that the label is now free.  If
possible, each Reclaim Ack message element should not be sent until
all data queued for transmission on the link, using the label
specified for release, has been sent.

If a Reclaim Ack message element is received specifying a flow for
which no Reclaim message element was issued, that Reclaim Ack message
element must be ignored, but no other Reclaim Ack message elements in
the same message must be affected.

If a Reclaim Ack message element is received specifying a different
label from the one sent in the original Reclaim message element for
that flow, the Reclaim Ack message element should be handled as if
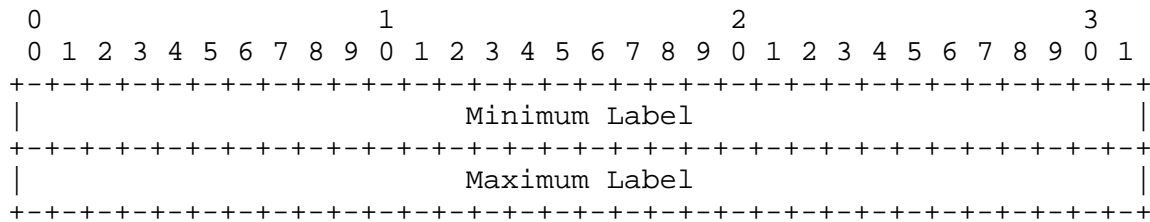the reclaim operation were successful.

If an error is detected in any of the fields of a Reclaim Ack message
element, that message element must be discarded, but no other Reclaim
Ack message elements in the same message must be affected.

The receiver should return an Error message to the sender only in the
case that the receiver does not understand the version of the IFMP
protocol in the received message or does not understand a Flow Type
in one of the Reclaim Ack message elements.

4.4 Label Range Message

The Label Range message element is sent in response to a Redirect
message if the label requested in one or more of the Redirect message
elements is outside the range that the receiver of the Redirect
message can handle.  The Label Range message informs the sender of
the Redirect message of the label range that can be handled on the
relevant link.

Only a single Label Range message element is permitted in a Label
Range message.  The Label Range message element has the following
structure:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Minimum Label                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Maximum Label                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```


Minimum Label
        The minimum value of label that can be specified in an IFMP
        Redirection Protocol message across this link.
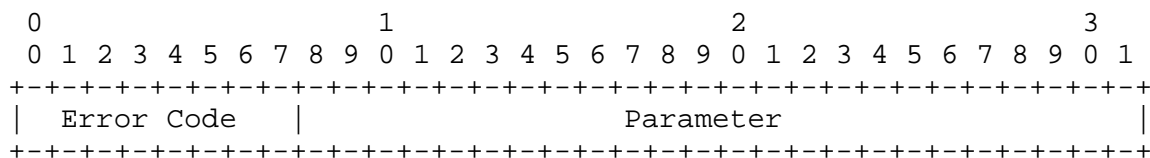
Maximum Label
        The maximum value of label that can be specified in an IFMP
        Redirection Protocol message across this link.

All values of label within the range Minimum Label to Maximum Label
inclusive may be specified in an IFMP Redirection Protocol message
across the link.

## 4.5 Error Message

An Error message can be sent by a node in response to any IFMP
Redirection Protocol message.

Only a single Error message element is permitted in an Error message.
The Error message element has the following structure:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Error Code  |                  Parameter                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```


Error Code
        Specifies which an error has occurred.

Each Error message can specify a single Parameter.

Two Error message elements are specified:


Bad Version:

Error Code = 1. The sender of the Error message cannot process the
            version of the IFMP protocol of the message that caused the
            error.  This message must only be sent if the version of
            the message that caused the error is greater than the most
            recent version that the sender of the Error message can
            process.  The parameter field of this Error message gives
            the most recent version of the IFMP protocol that the
            sender can process, right justified, with the unused most
            significant bits of the Parameter field set to zero.

Bad Flow Type:

Error Code = 2. The sender of the Error message does not understand a
            Flow Type that was received in the message that caused the
            error.  The Flow Type that caused the error is given in the
            parameter field, right justified, with the unused most
            significant bits of the Parameter field set to zero.

REFERENCES

   [ENCAP] Newman, P., et. al., "Transmission of Flow Labelled IPv4
           on ATM Data Links Ipsilon Version 1.0," Ipsilon Networks,
           RFC 1954, May 1996.

   [RFC793] Postel, J., "Transmission Control Protocol," STD 7, RFC
           793, September 1981.

SECURITY CONSIDERATIONS

   Security issues are not discussed in this memo.

AUTHORS' ADDRESSES

   Peter Newman                          Phone: +1 (415) 846-4603
   Ipsilon Networks, Inc.                EMail: pn@ipsilon.com

   W. L. Edwards, Chief Scientist        Phone:  +1 (913) 534 5334
   Sprint                                EMail:  texas@sprintcorp.com

   Robert M. Hinden                      Phone: +1 (415) 846-4604
   Ipsilon Networks, Inc.                EMail: hinden@ipsilon.com

   Eric Hoffman                          Phone: +1 (415) 846-4610
   Ipsilon Networks, Inc.                EMail: hoffman@ipsilon.com

   Fong Ching Liaw                       Phone: +1 (415) 846-4607
   Ipsilon Networks, Inc.                EMail: fong@ipsilon.com

   Tom Lyon                              Phone: +1 (415) 846-4601
   Ipsilon Networks, Inc.                EMail: pugs@ipsilon.com

   Greg Minshall                         Phone: +1 (415) 846-4605
   Ipsilon Networks, Inc.                EMail: minshall@ipsilon.com

Ipsilon Networks, Inc. is located at:

    2191 East Bayshore Road
    Suite 100
    Palo Alto, CA 94303
    USA

Sprint is located at:

    Sprint
    Sprint Technology Services - Long Distance Division
    9300 Metcalf Avenue
    Mailstop KSOPKB0802
    Overland Park, KS 66212-6333
    USA