Network Working Group
Request for Comments: 3077
Category: Standards Track

E. Duros
UDcast
W. Dabbous
INRIA Sophia-Antipolis
H. Izumiyama
N. Fujii
WIDE
Y. Zhang
HRL

March 2001

# A Link-Layer Tunneling Mechanism for Unidirectional Links

## Status of this Memo



## Copyright Notice



## Abstract

This document describes a mechanism to emulate full bidirectional connectivity between all nodes that are directly connected by a unidirectional link. The "receiver" uses a link-layer tunneling mechanism to forward datagrams to "feeds" over a separate bidirectional IP (Internet Protocol) network. As it is implemented at the link-layer, protocols in addition to IP may also be supported by this mechanism.

## 1. Introduction

Internet routing and upper layer protocols assume that links are bidirectional, i.e., directly connected hosts can communicate with each other over the same link.

This document describes a link-layer tunneling mechanism that allows a set of nodes (feeds and receivers, see Section 2 for terminology) which are directly connected by a unidirectional link to send datagrams as if they were all connected by a bidirectional link. We present a generic topology in section 3 with a tunneling mechanism

that supports multiple feeds and receivers. Note, this mechanism is not designed for topologies where a pair of nodes are connected by 2 unidirectional links in opposite direction.

The tunneling mechanism requires that all nodes have an additional interface to an IP interconnected infrastructure.

The tunneling mechanism is implemented at the link-layer of the interface of every node connected to the unidirectional link. The aim is to hide from higher layers, i.e., the network layer and above, the unidirectional nature of the link. The tunneling mechanism also includes an automatic tunnel configuration protocol that allows nodes to come up/down at any time.

Generic Routing Encapsulation [RFC2784] is suggested as the tunneling mechanism as it provides a means for carrying IP, ARP datagrams, and any other layer-3 protocol between nodes.

The tunneling mechanism described in this document was discussed and agreed upon by the UDLR working group.

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

## 2. Terminology

Unidirectional link (UDL): A one way transmission link, e.g., a broadcast satellite link.

Receiver: A router or a host that has receive-only connectivity to a UDL.

Send-only feed: A router that has send-only connectivity to a UDL.

Receive capable feed: A router that has send-and-receive connectivity to a UDL.

Feed: A send-only or a receive capable feed.

Node: A receiver or a feed.

Bidirectional interface: a typical communication interface that can send or receive packets, such as an Ethernet card, a modem, etc.

## 3. Topology

Feeds and receivers are connected via a unidirectional link. Send-only feeds can only send data over this unidirectional link, and receivers can only receive data from it. Receive capable feeds have both send and receive capabilities.

This mechanism has been designed to work with any topology with any number of receivers and one or more feeds. However, it is expected that the number of feeds will be small. In particular, the special case of a single send-only feed and multiple receivers is among the topologies supported.

A receiver has several interfaces, a receive-only interface and one or more additional bidirectional communication interfaces.

A feed has several interfaces, a send-only or a send-and-receive capable interface connected to the unidirectional link and one or more additional bidirectional communication interfaces. A feed MUST be a router.

Tunnels are constructed between the bidirectional interfaces of nodes, so these interfaces must be interconnected by an IP infrastructure. In this document we assume that that infrastructure is the Internet.

Figure 1 depicts a generic topology with several feeds and several receivers.

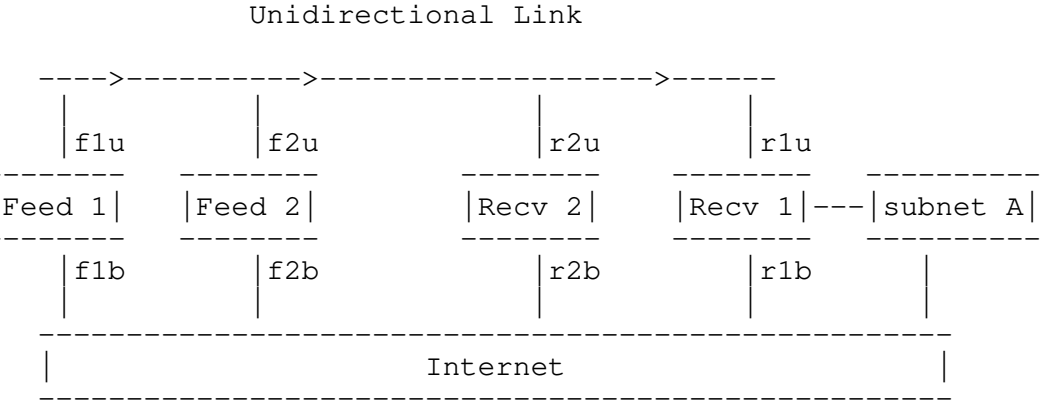



Figure 1: Generic topology

f1u (resp. f2u) is the IP address of the 'Feed 1' (resp. Feed 2) send-only interface.

f1b (resp. f2b) is the IP address of the 'Feed 1' (resp. Feed 2) bidirectional interface connected to the Internet.

r1u (resp. r2u) is the IP address of the 'Receiver 1' (resp. Receiver 2) receive-only interface.

r1b (resp. r2b) is the IP address of the 'Receiver 1' (resp. Receiver 2) bidirectional interface connected to the Internet.

Subnet A is a local area network connected to recv1.

Note that nodes have IP addresses on their unidirectional and their bidirectional interfaces. The addresses on the unidirectional interfaces (f1u, f2u, r1u, r2u) will be drawn from the same IP network. In general the addresses on the bidirectional interfaces (f1b, f2b, r1b, r2b) will be drawn from different IP networks, and the Internet will route between them.

## 4. Problems related to unidirectional links

Receive-only interfaces are "dumb" and send-only interfaces are "deaf". Thus a datagram passed to the link-layer driver of a receive-only interface is simply discarded. The link-layer of a send-only interface never receives anything.

The network layer has no knowledge of the underlying transmission technology except that it considers its access as bidirectional. Basically, for outgoing datagrams, the network layer selects the correct first hop on the connected network according to a routing table and passes the packet(s) to the appropriate link-layer driver.

Referring to Figure 1, Recv 1 and Feed 1 belong to the same network. However, if Recv 1 initiates a 'ping f1u', it cannot get a response from Feed 1. The network layer of Recv 1 delivers the packet to the driver of the receive-only interface, which obviously cannot send it to the feed.

Many protocols in the Internet assume that links are bidirectional. In particular, routing protocols used by directly connected routers no longer behave properly in the presence of a unidirectional link.

## 5. Emulating a broadcast bidirectional network

The simplest solution is to emulate a broadcast capable link-layer network. This will allow the immediate deployment of existing higher level protocols without change. Though other network structures, such as NBMA, could also be emulated, a broadcast network is more generally useful. Though a layer 3 network could be emulated, a

link-layer network allows the immediate use of any other network layer protocols, and most particularly allows the immediate use of ARP.

A link-layer tunneling mechanism which emulates bidirectional connectivity in the presence of a unidirectional link will be described in the next Section. We first consider the various communication scenarios which characterize a broadcast network in order to define what functionalities the link-layer tunneling mechanism has to perform in order to emulate a bidirectional broadcast link.

Here we enumerate the scenarios which would be feasible on a broadcast network, i.e., if feeds and receivers were connected by a bidirectional broadcast link:

Scenario 1: A receiver can send a packet to a feed (point-to-point communication between a receiver and a feed).

Scenario 2: A receiver can send a broadcast/multicast packet on the link to all nodes (point-to-multipoint).

Scenario 3: A receiver can send a packet to another receiver (point-to-point communication between two receivers).

Scenario 4: A feed can send a packet to a send-only feed (point-to-point communication between two feeds).

Scenario 5: A feed can send a broadcast/multicast packet on the link to all nodes (point-to-multipoint).

Scenario 6: A feed can send a packet to a receiver or a receive capable feed (point-to-point).

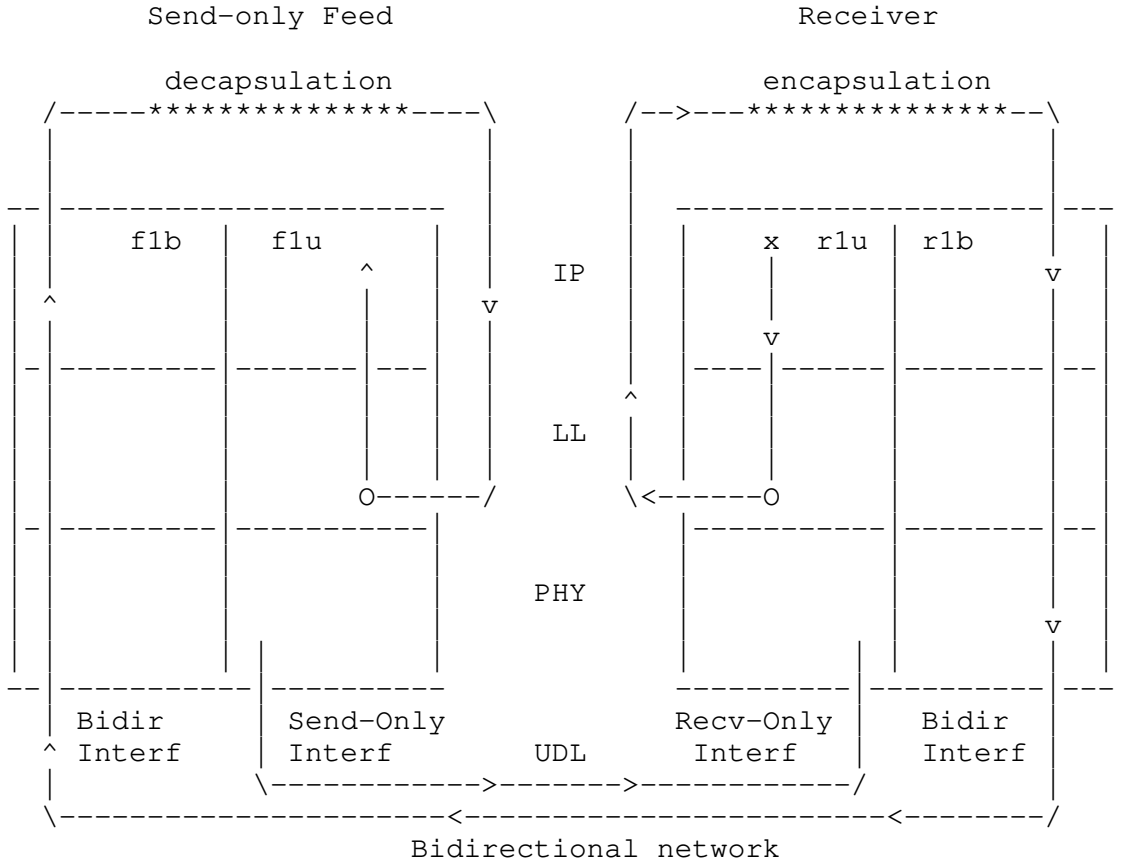
These scenarios are possible on a broadcast network. Scenario 6 is already feasible on the unidirectional link. The link-layer tunneling mechanism should therefore provide the functionality to support scenarios 1 to 5.

Note that regular IP forwarding over such an emulated network (i.e., using the emulated network as a transit network) works correctly; the next hop address at the receiver will be the unidirectional link address of another router (a feed or a receiver) which will then relay the packet.

## 6. Link-layer tunneling mechanism

This link-layer tunneling mechanism operates underneath the network layer. Its aim is to emulate bidirectional link-layer connectivity. This is transparent to the network layer: the link appears and behaves to the network layer as if it was bidirectional.

Figure 2 depicts a layered representation of the link-layer tunneling mechanism in the case of Scenario 1.




x : IP layer at the receiver generates a datagram to be forwarded on the receive-only interface.
O : Entry point where the link-layer tunneling mechanism is triggered.

Figure 2: Scenario 1 using the link-layer Tunneling Mechanism

### 6.1. Tunneling mechanism on the receiver

On the receiver, a datagram is delivered to the link-layer of the unidirectional interface for transmission (see Figure 2). It is then encapsulated within a MAC header corresponding to the unidirectional link. This packet cannot be sent directly over the link, so it is then processed by the tunneling mechanism.

The packet is encapsulated within an IP header whose destination is the IP address of a feed bidirectional interface (f1b or f2b). This destination address is also called the tunnel end-point. The mechanism for a receiver to learn these addresses and to choose the feed is explained in Section 7. The type of encapsulation is described in Section 8.

In all cases the packet is encapsulated, but the tunnel end-point (an IP address) depends on the encapsulated packet's destination MAC address. If the destination MAC address is:

1) the MAC address of a feed interface connected to the unidirectional link (Scenario 1). The datagram is encapsulated, the destination address of the encapsulating datagram is the feed tunnel end-point (f1b referring to Figure 2).

2) a MAC broadcast/multicast address (Scenario 2). The datagram is encapsulated, the destination address of the encapsulating datagram is the default feed tunnel end-point. See Section 7.4 for further details on the default feed.

3) a MAC address that belongs to the unidirectional network but is not a feed address (Scenario 3). The datagram is encapsulated, the destination address of the encapsulating datagram is the default feed tunnel end-point.

The encapsulated datagram is passed to the network layer which forwards it according to its destination address. The destination address is a feed bidirectional interface which is reachable via the Internet. In this case, the encapsulated datagram is forwarded via the receiver bidirectional interface (r1b).

### 6.2. Tunneling mechanism on the feed

A feed processes unidirectional link related packets in two different ways:

- packets generated by a local application or packets routed as usual by the IP layer may have to be forwarded over the unidirectional link (Section 6.2.1)

- encapsulated packets received from another receiver or feed need tunnel processing (Section 6.2.2).

A feed cannot directly send a packet to a send-only feed over the unidirectional link (Scenario 4). In order to emulate this type of communication, feeds have to tunnel packets to send-only feeds. A feed MUST maintain a list of all other feed tunnel end-points. This list MUST indicate which are send-only feed tunnel end-points. This is configured manually at the feed by the local administrator, as described in Section 7.

### 6.2.1. Forwarding packets over the unidirectional link

When a datagram is delivered to the link-layer of the unidirectional interface of a feed for transmission, its treatment depends on the packet's destination MAC address. If the destination MAC address is:

1) the MAC address of a receiver or a receive capable feed (Scenario 6). The packet is sent over the unidirectional link. This is classical "forwarding".

2) the MAC address of a send-only feed (Scenario 4). The packet is encapsulated and sent to the send-only feed tunnel end-point. The type of encapsulation is described in Section 8.

3) a broadcast/multicast destination (Scenario 5). The packet is sent over the unidirectional link. Concurrently, a copy of this packet is encapsulated and sent to every feed of the list of send-only feed tunnel end-points. Thus the broadcast/multicast will reach all receivers and all send-only feeds.

### 6.2.2. Receiving encapsulated packets

Feeds listen for incoming encapsulated datagrams on their tunnel end-points. Encapsulated packets will have been received on a bidirectional interface, and traversed their way up the IP stack. They will then enter a decapsulation process (See Figure 2).

Decapsulation reveals the original link-layer packet. Note that this has not been modified in any way by intermediate routers; in particular, the original MAC header will be intact.

Further actions depend on the destination MAC address of the link-layer packet, which can be:

1) the MAC address of the feed interface connected to the unidirectional link, i.e., own MAC address (Scenarios 1 and 4). The packet is passed to the link-layer of the interface connected to the unidirectional link which can then deliver it up to higher layers. As a result, the datagram is processed as if it was coming from the unidirectional link, and being delivered locally. Scenarios 1 and 4 are now feasible, a receiver or a feed can send a packet to a feed.

2) a receiver address (Scenario 3). The packet is passed to the link-layer of the interface connected to the unidirectional link. It is directly sent over the unidirectional link, to the indicated receiver. Note, the packet must not be delivered locally. Scenario 3 is now feasible, a receiver can send a packet to another receiver.

3) a broadcast/multicast address, this corresponds to Scenarios 2 and 5. We have to distinguish two cases, either (i) the encapsulated packet was sent from a receiver or (ii) from a feed (encapsulated broadcast/multicast packet sent to a send-only feed). These cases are distinguished by examining the source address of the encapsulating packet and comparing it with the configured list of feed IP addresses. The action then taken is:

   i) the feed was designated as a default feed by a receiver to forward the broadcast/multicast packet. The feed is then in charge of sending the multicast packet to all nodes. Delivery to all nodes is accomplished by executing all 3 of the following actions:

      - The packet is encapsulated and sent to the list of send-only feed tunnel end-points.
      - Also, the packet is passed to the link-layer of the interface which forwards it directly over the unidirectional link (all receivers and receive capable feeds receive it).
      - Also, the link-layer delivers it locally to higher layers.

      Caution: a receiver which sends an encapsulated broadcast/multicast packet to a default feed will receive its own packet via the unidirectional link. Correct filtering as described in [RFC1112] must be applied.

ii) the feed receives the packet and keeps it for local delivery. The packet is passed to the link-layer of the interface connected to the unidirectional link which delivers it to higher layers.

Scenario 2 is now feasible, a receiver can send a broadcast/multicast packet over the unidirectional link and it will be heard by all nodes.

## 7. Dynamic Tunnel Configuration Protocol (DTCP)

Receivers and feeds have to know the feed tunnel end-points in order to forward encapsulated datagrams (e.g., Scenarios 1 and 4).

The number of feeds is expected to be relatively small (Section 3), so at every feed the list of all feeds is configured manually. This list should note which are send-only feeds, and which are receive capable feeds. The administrator sets up tunnels to all send-only feeds. A tunnel end-point is an IP address of a bidirectional link on a send-only feed.

For scalability reasons, manual configuration cannot be done at the receivers. Tunnels must be configured and maintained dynamically by receivers, both for scalability, and in order to cope with the following events:

1) New feed detection.
   When a new feed comes up, every receiver must create a tunnel to enable bidirectional communication with it.

2) Loss of unidirectional link detection.
   When the unidirectional link is down, receivers must disable their tunnels. The tunneling mechanism emulates bidirectional connectivity between nodes. Therefore, if the unidirectional link is down, a feed should not receive datagrams from the receivers. Protocols that consider a link as operational if they receive datagrams from it (e.g., the RIP protocol [RFC2453]) require this behavior for correct operation.

3) Loss of feed detection.
   When a feed is down, receivers must disable their corresponding tunnel. This prevents unnecessary datagrams from being tunneled which might overload the Internet. For instance, there is no need for receivers to forward a broadcast message through a tunnel whose end-point is down.

The DTCP protocol provides a means for receivers to dynamically discover the presence of feeds and to maintain a list of operational tunnel end-points. Feeds periodically announce their tunnel end-point addresses over the unidirectional link. Receivers listen to these announcements and maintain a list of tunnel end-points.

### 7.1. The HELLO message

The DTCP protocol is a 'unidirectional protocol', messages are only sent from feeds to receivers.

The packet format is shown in Figure 3. Fields contain binary integers, in normal Internet order with the most significant bit first. Each tick mark represents one bit.

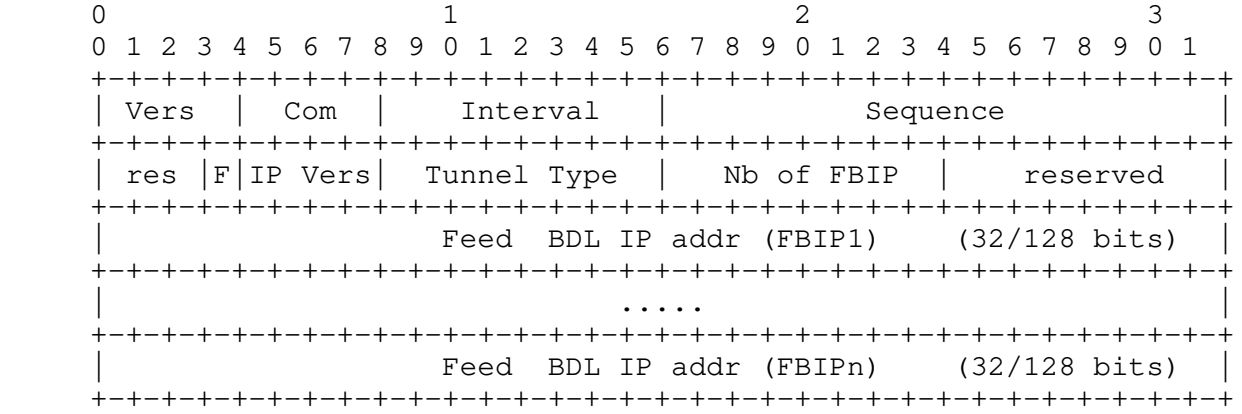



Figure 3: Packet Format

Every datagram contains the following fields, note that constants are written in uppercase and are defined in Section 7.5:

Vers (4 bit unsigned integer): DTCP version number. MUST be DTCP_VERSION.

Com (4 bit unsigned integer): Command field, possible values are
- 1 - JOIN A message announcing that the feed sending this message is up and running.
- 2 - LEAVE A message announcing that the feed sending this message is being shut down.

Interval (8 bit unsigned integer): Interval in seconds between HELLO messages for the IP protocol in "IP Vers". Must be > 0. The recommended value is HELLO_INTERVAL. If this value is increased, the feed MUST continue to send HELLO messages at the old rate for at least the old HELLO_LEAVE period.

Sequence (16 bit unsigned integer): Random value initialized at boot time and incremented by 1 every time a value of the HELLO message is modified.

res (3 bits): Reserved/unused field, MUST be zero.

F (1 bit): bit indicating the type of feed:
0 = Send-only feed
1 = Receive-capable feed

IP Vers (4 bit unsigned integer): IP protocol version of the feed bidirectional IP addresses (FBIP):
4 = IP version 4
6 = IP version 6

Tunnel Type (8 bit unsigned integer): tunneling protocol supported by the feed. This value is the IP protocol number defined in [RFC1700] [iana/protocol-numbers] and their legitimate descendents. Receivers MUST use this form of tunnel encapsulation when tunneling to the feed.
47 = GRE [RFC2784] (recommended)
Other protocol types allowing link-layer encapsulation are permitted. Obtaining new values is documented in [RFC2780].

Nb of FBIP (8 bit unsigned integer): Number of bidirectional IP feed addresses which are enumerated in the HELLO message

reserved (8 bits): Reserved/unused field, MUST be zero.

Feed BDL IP addr (32 or 128 bits). The bidirectional IP address feed is the IP address of a feed bidirectional interface (tunnel end-point) reachable via the Internet. A feed has 'Nb of FBIP' IP addresses which are operational tunnel end-points. They are enumerated in preferred order. FBIP1 being the most suitable tunnel end-point.

### 7.2. DTCP on the feed: sending HELLO packets

The DTCP protocol runs on top of UDP. Packets are sent to the "DTCP announcement" multicast address over the unidirectional link on port HELLO_PORT with a TTL of 1. Due to existing deployments a feed SHOULD also support the use of the old DTCP announcement address, as described in Appendix B.

The source address of the HELLO packet is set to the IP address of the feed interface connected to the unidirectional link. In the rest of the document, this value is called FUIP (Feed Unidirectional IP address).

The process in charge of sending HELLO packets fills every field of the datagram according to the description given in Section 7.1.

As long as a feed is up and running, it periodically announces its presence to receivers. It MUST send HELLO packets containing a JOIN command every HELLO_INTERVAL over the unidirectional link.

Referring to Figure 1 in Section 3, Feed 1 (resp. Feed 2) sends HELLO messages with the FBIP1 field set to f1b (resp. f2b).

When a feed is about to be shut down, or when routing over the unidirectional link is about to be intentionally interrupted, it is recommended that feeds:

1) stop sending HELLO messages containing a JOIN command.

2) send a HELLO message containing a LEAVE command to inform receivers that the feed is no longer performing routing over the unidirectional link.

### 7.3. DTCP on the receiver: receiving HELLO packets

Based on the reception of HELLO messages, receivers discover the presence of feeds, maintain a list of active feeds, and keep track of the tunnel end-points for those feeds.

For each active feed, and each IP protocol supported, at least the following information will be kept:

| | |
|---|---|
| FUIP | - feed unidirectional link IP address |
| FUMAC | - MAC address corresponding to the above IP address |
| (FBIP1,...,FBIPn) | - list of tunnel end-points |
| tunnel type | - tunnel type supported by this feed |
| Sequence | - "Sequence" value from the last HELLO received from this feed |
| timer | - used to timeout this entry |

The FUMAC value for an active feed is needed for the operation of this protocol. However, the method of discovery of this value is not specified here.

Initially, the list of active feeds is empty.

When a receiver is started, it MUST run a process which joins the "DTCP announcement" multicast group and listens to incoming packets on the HELLO_PORT port from the unidirectional link.

Upon the reception of a HELLO message, the process checks the version number of the protocol. If it is different from HELLO_VERSION, the packet is discarded and the process waits for the next incoming packet.

After successfully checking the version number further action depends on the type of command:

- JOIN:

  The process verifies if the address FUIP already belongs to the list of active feeds.

  If it does not, a new entry, for feed FUIP, is created and added to the list of active feeds. The number of feed bidirectional IP addresses to read is deduced from the 'Nb of FBID' field. These tunnel end-points (FBIP1,...,FBIPn) can then be added to the new entry. The tunnel Type and Sequence values are also taken from the HELLO packet and recorded in the new entry. A timer set to HELLO_LEAVE is associated with this entry.

  If it does, the sequence number is compared to the sequence number contained in the previous HELLO packet sent by this feed. If they are equal, the timer associated with this entry is reset to HELLO_LEAVE. Otherwise all the information corresponding to FUIP is set to the values from the HELLO packet.

  Referring to Figure 1 in Section 3, both receivers (recv 1 and recv 2) have a list of active feeds containing two entries: Feed 1 with a FUIP of f1u and a list of tunnel end-points (f1b); and Feed 2 with a FUIP of f2u and a list of tunnel end-points (f2b).

- LEAVE:

  The process checks if there is an entry for FUIP in the list of active feeds. If there is, the timer is disabled and the entry is deleted from the list. The LEAVE message provides a means of quickly updating the list of active feeds.

A timeout occurs for either of two reasons:

1) a feed went down without sending a LEAVE message. As JOIN messages are no longer sent from this feed, a timeout occurs at HELLO_LEAVE after the last JOIN message.

2) the unidirectional link is down. Thus no more JOIN messages are received from any of the feeds, and they will each timeout independently. The timeout of each entry depends on its

individual HELLO_LEAVE value, and when the last JOIN message was sent by that feed, before the unidirectional link went down.

In either case, bidirectional connectivity can no longer be ensured between the receiver and the feed (FUIP): either the feed is no longer routing datagrams over the unidirectional link, or the link is down. Thus the associated entry is removed from the list of active feeds, whatever the cause. As a result, the list only contains operational tunnel end-points.

The HELLO protocol provides receivers with a list of feeds, and a list of usable tunnel end-points (FBIP1,..., FBIPn) for each feed. In the following Section, we describe how to integrate the HELLO protocol into the tunneling mechanism described in Sections 6.1 and 6.2.

### 7.4. Tunneling mechanism using the list of active feeds

This Section explains how the tunneling mechanism uses the list of active feeds to handle datagrams which are to be tunneled. Referring to Section 6.1, it shows how feed tunnel end-points are selected.

The choice of the default feed is made independently at each receiver. The choice is a matter of local policy, and this policy is out of scope for this document. However, as an example, the default feed may be the feed that has the lowest round trip time to the receiver.

When a receiver sends a packet to a feed, it must choose a tunnel end-point from within the FBIP list. The 'preferred FBIP' is generally FBIP1 (Section 7.1). For various reasons, a receiver may decide to use a different FBIP, say FBIPi instead of FBIP1, as the tunnel end-point. For example, the receiver may have better connectivity to FBIPi. This decision is taken by the receiver administrator.

Here we show how the list of active feeds is involved when a receiver tunnels a link-layer packet. Section 6.1 listed the following cases, depending on whether the MAC destination address of the packet is:

1) the MAC address of a feed interface connected to the unidirectional link: This is TRUE if the address matches a FUMAC address in the list of active feeds. The packet is tunneled to the preferred FBIP of the matching feed.

2) the broadcast address of the unidirectional link or a multicast address:

This is determined by the MAC address format rules, and the list of active feeds is not involved. The packet is tunneled to the preferred FBIP of the default feed.

3) an address that belongs to the unidirectional network but is not a feed address:
This is TRUE if the address is neither broadcast nor multicast, nor found in the list of active feeds. The packet is tunneled to the preferred FBIP of the default feed.

In all cases, the encapsulation type depends on the tunnel type required by the feed which is selected.

### 7.5. Constant definitions

DTCP_VERSION is 1.

HELLO_INTERVAL is 5 seconds.

"DTCP announcement" multicast group is 224.0.0.36, assigned by IANA.

HELLO_PORT is 652. It is a reserved system port assigned by IANA, no other traffic must be allowed.

HELLO_LEAVE is 3*Interval, as advertised in a HELLO packet, i.e., 15 seconds if the default HELLO_INTERVAL was advertised.

## 8. Tunnel encapsulation format

The tunneling mechanism operates at the link-layer and emulates bidirectional connectivity amongst receivers and feeds. We assume that hardware connected to the unidirectional link supports broadcast and unicast MAC addressing. That is, a feed can send a packet to a particular receiver using a unicast MAC destination address or to a set of receivers using a broadcast/multicast destination address. The hardware (or the driver) of the receiver can then filter the incoming packets sent over the unidirectional links without any assumption about the encapsulated data type.

In a similar way, a receiver should be capable of sending unicast and broadcast MAC packets via its tunnels. Link-layer packets are encapsulated. As a result, after decapsulating an incoming packet, the feed can perform link-layer filtering as if the data came directly from the unidirectional link (See Figure 2).

Generic Routing Encapsulation (GRE) [RFC2784] suits our requirements because it specifies a protocol for encapsulating arbitrary packets, and allows use of IP as the delivery protocol.

The feed's local administrator decides what encapsulation it will demand that receivers use, and sets the tunnel type field in the HELLO message appropriately. The value 47 (decimal) indicates GRE. Other values can be used, but their interpretation must be agreed upon between feeds and receivers. Such usage is not defined here.

### 8.1. Generic Routing Encapsulation on the receiver

A GRE packet is composed of a header in which a type field specifies the encapsulated protocol (ARP, IP, IPX, etc.). See [RFC2784] for details about the encapsulation. In our case, only support for the MAC addressing scheme of the unidirectional link MUST be implemented.

A packet tunneled with a GRE encapsulation has the following format: the delivery header is an IP header whose destination is the tunnel end-point (FBIP), followed by a GRE header specifying the link-layer type of the unidirectional link. Figure 4 presents the entire encapsulated packet.

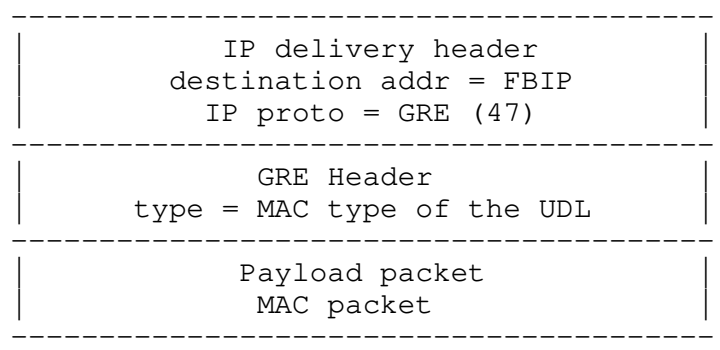



Figure 4: Encapsulated packet

## 9. Issues

### 9.1. Hardware address resolution

Regardless of whether the link is unidirectional or bidirectional, if a feed sends a packet over a non-point-to-point type network, it requires the data link address of the destination. ARP [RFC826] is used on Ethernet networks for this purpose.

The link-layer mechanism emulates a bidirectional network in the presence of an unidirectional link. However, there are asymmetric delays between every (feed, receiver) pair. The backchannel between a receiver and a feed has varying delays because packets go through the Internet. Furthermore, a typical example of a unidirectional link is a GEO satellite link whose delay is about 250 milliseconds.

Because of long round trip delays, reactive address resolution methods such as ARP [RFC826] may not work well. For example, a feed may have to forward packets at high data rates to a receiver whose hardware address is unknown. The stream of packets is passed to the link-layer driver of the feed send-only interface. When the first packet arrives, the link-layer realizes it does not have the corresponding hardware address of the next hop, and sends an ARP request. While the link-layer is waiting for the response (at least 250 ms for the GEO satellite case), IP packets are buffered by the feed. If it runs out of space before the ARP response arrives, IP packets will be dropped.

This problem of address resolution protocols is not addressed in this document. An ad-hoc solution is possible when the MAC address is configurable, which is possible in some satellite receiver cards. A simple transformation (maybe null) of the IP address can then be used as the MAC address. In this case, senders do not need to "resolve" an IP address to a MAC address, they just need to perform the simple transformation.

### 9.2. Routing protocols

The link-layer tunneling mechanism hides from the network and higher layers the fact that feeds and receivers are connected by a unidirectional link. Communication is bidirectional, but asymmetric in bandwidths and delays.

In order to incorporate unidirectional links in the Internet, feeds and receivers might have to run routing protocols in some topologies. These protocols will work fine because the tunneling mechanism results in bidirectional connectivity between all feeds and receivers. Thus routing messages can be exchanged as on any bidirectional network.

The tunneling mechanism allows any IP traffic, not just routing protocol messages, to be forwarded between receivers and feeds. Receivers can route datagrams on the Internet using the most suitable feed or receiver as a next hop. Administrators may want to set the metrics used by their routing protocols in order to reflect in routing tables the asymmetric characteristics of the link, and thus direct traffic over appropriate paths.

Feeds and receivers may implement multicast routing and therefore dynamic multicast routing can be performed over the unidirectional link. However issues related to multicast routing (e.g., protocol configuration) are not addressed in this document.

### 9.3. Scalability

The DTCP protocol does not generate a lot of traffic whatever the number of nodes. The problem with a large number of nodes is not related to this protocol but to more general issues such as the maximum number of nodes which can be connected to any link. This is out of scope of this document.

## 10. IANA Considerations

IANA has reserved the address 224.0.0.36 for the "DTCP announcement" multicast address as defined in Section 7.

IANA has reserved the udp port 652 for the HELLO_PORT as defined in Section 7.

## 11. Security Considerations

Many unidirectional link technologies are characterised by the ease with which the link contents can be received. If sensitive or valuable information is being sent, then link-layer security mechanisms are an appropriate measure. For the UDLR protocol itself, the feed tunnel end-point addresses, sent in HELLO messages, may be considered sensitive. In such cases link-layer security mechanisms may be used.

Security in a network using the link-layer tunneling mechanism should be relatively similar to security in a normal IPv4 network. However, as the link-layer tunneling mechanism requires the use of tunnels, it introduces a potential for unauthorised access to the service. In particular, ARP and IP spoofing are potential threats because nodes may not be authorised to tunnel packets. This can be countered by authenticating all tunnels. The authenticating mechanism is not specified in this document, it can take place either in the delivery IP protocol (e.g., AH[RFC2402]) or in an authentication protocol integrated with the tunneling mechanism.

At a higher level, receivers may not be authorised to provide routing information even though they are connected to the unidirectional link. In order to prevent unauthorised receivers from providing fake routing information, routing protocols running on top of the link-layer tunneling mechanism MUST use authentication mechanisms when available.

## 12. Acknowledgments

We would like to thank Tim Gleeson (Cisco Japan) for his valuable editing and technical input during the finalization phase of the document.

We would like to thank Patrick Cipiere (UDcast) for his valuable input concerning the design of the encapsulation mechanism.

We would like also to thank for their participation: Akihiro Tosaka (IMD), Akira Kato (Tokyo Univ.), Hitoshi Asaeda (IBM/ITS), Hiromi Komatsu (JSAT), Hiroyuki Kusumoto (Keio Univ.), Kazuhiro Hara (Sony), Kenji Fujisawa (Sony), Mikiyo Nishida (Keio Univ.), Noritoshi Demizu (Sony CSL), Jun Murai (Keio Univ.), Jun Takei (JSAT) and Harri Hakulinen (Nokia).

## Appendix A: Conformance and interoperability

This document describes a mechanism to emulate bidirectional connectivity between nodes that are directly connected by a unidirectional link. Applicability over a variety of equipment and environments is ensured by allowing a choice of several key system parameters.

Thus in order to ensure interoperability of equipment it is not enough to simply claim conformance with the mechanism defined here. A usage profile for a particular environment will require the definition of several parameters:

- the MAC format used
- the tunneling mechanism to be used (GRE is recommended)
- the "tunnel type" indication if GRE is not used

For example, a system might claim to implement "the link-layer tunneling mechanism for unidirectional links, using IEEE 802 LLC, and GRE encapsulation for the tunnels."

## Appendix B: DTCP announcement address transition plan

Some older receivers listen for DTCP announcements on the 224.0.1.124 multicast address (the "old DTCP announcement" address). In order to support such legacy receivers, feeds SHOULD be configurable to send all announcements simultaneously to both the "DTCP announcement" address, and the "old DTCP announcement" address. The default setting is to send announcements to just the "DTCP announcement" address.

In order to encourage the transition plan, the "old" feeds MUST be updated to send DTCP announcements as defined in this section. The number of "old" feeds originally deployed is relatively small and therefore the update should be fairly easy. "New" receivers only support "new" feeds, i.e., they listen to DTCP announcements on the "DTCP announcement" address.

References

[RFC826] Plummer, D., "An Ethernet Address Resolution Protocol", STD 37, RFC 826, November 1982.

[RFC1112] Deering, S., "Host Extensions for IP Multicasting", STD 5, RFC 1112, August 1989

[RFC1700] Reynolds, J. and J. Postel, "ASSIGNED NUMBERS", STD 2, RFC 1700, October 1994.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2402] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[RFC2453] Malkin, G., "RIP Version 2", STD 56, RFC 2453, November 1998.

[RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, March 2000.

[RFC2784] Farinacci, D., Hanks, S., Meyer, D. and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

Authors' Addresses

Emmanuel Duros
UDcast
1681, route des Dolines
Les Taissounieres - BP 355
06906 Sophia-Antipolis Cedex
France

Phone : +33 4 93 00 16 60
Fax : +33 4 93 00 16 61
EMail : Emmanuel.Duros@UDcast.com

Walid Dabbous
INRIA Sophia Antipolis
2004, Route des Lucioles BP 93
06902 Sophia Antipolis
France

Phone : +33 4 92 38 77 18
Fax : +33 4 92 38 79 78
EMail : Walid.Dabbous@inria.fr

Hidetaka Izumiyama
JSAT Corporation
Toranomon 17 Mori Bldg.5F
1-26-5 Toranomon, Minato-ku
Tokyo 105
Japan

Phone : +81-3-5511-7568
Fax : +81-3-5512-7181
EMail : izu@jsat.net

Noboru Fujii
Sony Corporation
2-10-14 Osaki, Shinagawa-ku
Tokyo 141
Japan

Phone : +81-3-3495-3092
Fax : +81-3-3495-3527
EMail : fujii@dct.sony.co.jp

Yongguang Zhang
HRL
RL-96, 3011 Malibu Canyon Road
Malibu, CA 90265,
USA

Phone : 310-317-5147
Fax : 310-317-5695
EMail : ygz@hrl.com

## Full Copyright Statement



## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.