Dynamic Routing in Networks with Unidirectional Links

Walid Dabbous Emmanuel Duros Thierry Ernst*

INRIA Sophia Antipolis†
2004, route des Lucioles - BP 93
06902 Sophia-Antipolis, France
{dabbous|eduros|ternst}@sophia.inria.fr

Abstract

Current routing protocols assume that routers are connected by bidirectional links. However, in an increasing number of configurations, pairs of routers may be connected by unidirectional links. Existing routing protocols do not work properly in such configuration. A particular example is GEO satellite links with receive-only hardware and a back channel through a bidirectional network. In this paper we present two approaches to allow dynamic routing in this configuration. One is based on routing protocol modification and the other on tunneling, both currently discussed within the IETF UDLR working group.

More general examples include networks where several links can be unidirectional. Neither routing protocol modifications nor tunneling are applicable in such configuration. We have therefore designed a new protocol based on circuit discovery. We describe this protocol which allows dynamic routing in networks where all links are unidirectional.

1 Introduction

Current routing protocols assume that pairs of routers are connected by symmetric links. Symmetry means that the links are bidirectional and have the same bandwidth and routing metric in each direction. Decisions concerning reachability and metric calculation are based on this link symmetry assumption.

Recently, new transmission media offer asymmetric and even unidirectional links. Unidirectional links, i.e. links with zero return bandwidth, are experienced in several configurations namely 1) GEO satellite links with receive-only hardware, 2) ad-hoc wireless networks where pairs of routers do not have the same transmitting power, 3) unidirectional cables, etc.

Because current routing protocol assume that links are bidirectional, routers connected to an "outgoing" unidirectional link cannot be aware of the existence of networks reachable through this link. However, the problem is more or less hard to solve according to the considered configuration.

GEO satellite links providing receive-only connectivity to the Internet (with a terrestrial back channel) are a particular configuration of unidirectional links. This configuration is interesting since satellites offer a high bandwidth transmission and a large geographical coverage. In addition, their broadcast nature could be very useful for multicasting purposes. They can as well bypass bottlenecks by providing high bandwidth links over large geographical areas. Those advantages are of great interest to provide Internet access. However, the cost of VSAT antennas - the devices allowing emission of data toward satellites, is too high for their use on a large scale at the moment. On the other hand, the cost of receive-only hardware is negligible. Broadcast satellite links may therefore be used to provide "alternative" high speed receive-only access to the Internet, with the return traffic sent through a low speed modem access or any other Internet connection. This configuration is ideal for one-to-many multicast applications, such as web site mirroring, seminar multicasting, video-on-demand, etc. The problem here is the following: how to perform dynamic routing in a net-

*Authors in alphabetic order

†This work is supported in part by Eutelsat, http://www.eutelsat.org

work where unidirectional links are “alternative” links on top of a bidirectional network providing a back channel from the receive-only router to the send-only router connected to the unidirectional link.

We raised the problem of “alternative” unidirectional link on top of a bidirectional network for discussion within the IETF UDLR working group launched for this purpose. The objective of the working group is to design a “short term” solution to this specific problem. We worked on two approaches to solve the problem. The first one is based on routing protocol modification and the other on tunneling. We are experimenting both approaches on a local test network and soon on a satellite link.

There are other configurations where there is no bidirectional back channel (e.g. the back channel itself comprises a unidirectional link). The network is represented by a directed graph in this case. If each node of this graph can reach every other node, bidirectional communication is feasible even if several links between the source and the destination are unidirectional. Neither routing protocol modifications nor tunneling are applicable; an adequate mechanism is needed in order to perform dynamic routing in this general case.

Then, we have been working on the design of a so-called “long term” solution where no link bidirectionality assumptions are made. We designed a circuit-based routing protocol that allows dynamic routing in a network where all links are unidirectional.

We present in this paper a general description of our work in this domain, focused on dynamic routing issues in networks with unidirectional links. We did not consider transmission control problems nor quality of service issues. The rest of the paper is structured as follows. Section 2 presents the terminology used. Section 3 details the routing issues for both short term and long term cases. Section 4 presents and discusses the two approaches for the short term solution, and provides some experimentation results. Section 5 explains the need for a new protocol to provide dynamic routing in the general case, and presents briefly a protocol that we designed for this purpose. Section 6 concludes the paper and presents future work.

2 Terminology

We define here terms that we use in this paper. Note that the terminology is being discussed in the UDLR Working Group.

1. **Unidirectional link**: A zero return bandwidth link.
2. **Feed**: A router connected to an unidirectional link with a send-only interface.
3. **Receiver**: A router connected to an unidirectional link with a receive-only interface.
4. **Bidirectional network**: A network where all links are bidirectional.
5. **Unidirectional network**: A network where all links are unidirectional.
6. **Mixed bidirectional/unidirectional**: Network also called Mixed network.
7. **Bidirectional path**: A succession of bidirectional links.
8. **Back channel**: A bidirectional path between a receiver and a feed.

3 Routing issues

A satellite network comprises two types of stations: feeds and receivers. A receiver is composed of a satellite dish oriented toward a geostationary satellite, connected via an interface to a router. The router has one or more extra interfaces, connected to the Internet. Figure 1 shows the architecture of a topology using a satellite link.

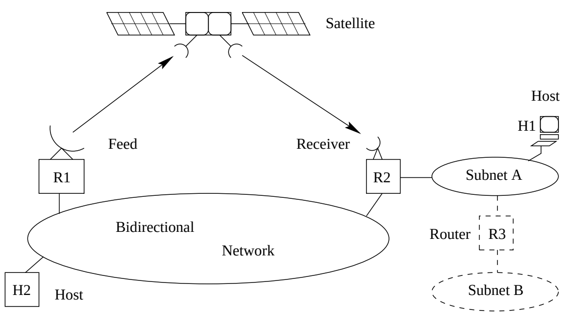

Figure 1: unidirectional overlays on top of bidirectional network

In this example, let's assume that host *H1* connected to *Subnet A* wants to initiate a communication with host *H2*. *H1* sends its requests to its default router *R2*. *R2* learnt with the use of routing protocols how to get to the destination *H2* and forwards the datagrams to the next hop that belongs to the shortest path between *R2* and *H2*. As *H2* receives *H1* requests, we expect *H2* to send its replies via the satellite network assuming that it belongs to the shortest path between *H2* and *H1*. Feeds cover nationwide areas and are supposed to be installed at strategic positions in order to create shorter paths over the Internet.

Figure 2 gives a more detailed view of the topology. The routers named *Rx* are classical IP routers connected to bidirectional links (optical fibers, coaxial cables, etc.). *R1* is a feed and *R2* is a receiver.

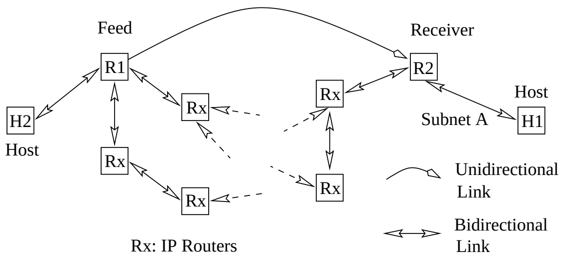

Figure 2: Detailed view of the topology

It is obvious that the route from *H2* to *H1* via the satellite network should be preferred to the route via terrestrial links as it is the shortest path. In this example, packets are forwarded by two hops whereas via terrestrial links it would require to pass through a more important number of hops. However, in order to route *H2* packets to *H1* via the unidirectional link, *R1* must learn that *H1* is reachable via the unidirectional link.

The main problem with unidirectional links is that common routing protocols such as RIP [9], OSPF [11] and DVMRP [16] do not support this type of links. Indeed, adjacent routers exchange their routing information to create forwarding tables. For instance, routers broadcast RIP messages on all their "directly connected subnetworks". In Figure 2, the receiver must announce to the feed that *Subnet A* is reachable via the unidirectional link. Then, the feed knows it can forward packets coming from *H2* to *Subnet A* via the unidirectional link. However, due to the nature of the link between the feed and the receiver, *R2* cannot inform *R1* of the presence of *Subnet A* and then *R1* will never use the unidirectional link to forward packets.

There are two possibilities concerning the "learning method", the first one is to define statically routes to destinations in feeds, the second one is to use dynamic routing protocols.

3.1 Static routing

To overcome the routing problem cited above, it is technically possible to substitute the use of routing protocol by manually setting up routes in feeds. Referring to Figure 2, we can add a route in *R1* which would be *"to reach Subnet A, send packet to R2 via the unidirectional link"*. This way, packets sent by *H2* to *H1* are automatically forwarded over the unidirectional link.

This solution works fine but is not scalable and does not take into account possible breakdowns:

- Satellite are capable of covering large geographical areas and therefore potentially address a large set of receivers. These receivers are connected to subnetworks which are probably themselves interconnected with other subnetworks (*Subnet B* in Figure 1). Therefore it would be irrelevant to set up manually routes to hundreds or thousands networks reachable via various receivers.
- satellite links are like any other links, they are expected to work 100% of the time unless: someone powers off the antenna, someone accidentally (or intentionally) interferes with the emission frequency, someone wants to see what is going on inside the feed horn, etc. If any of these events occurs, packets are not transmitted over the satellite link and we then create a black-hole.

A static approach is therefore not recommended and we will prefer a dynamic one which figures out the previous problems.

3.2 Dynamic routing

Unlike static routing, routing protocols provide a way of learning routes dynamically. Whatever the number of subnetworks which might be connected to the receivers, feeds would learn automatically their presence. Furthermore, if unexpected events happen and a network gets down, routers directly connected to it inform their neighbor routers and the information is propagated to the rest of the autonomous system. Routes are then modified in order to isolate the network which is down.

Dynamic routing is obviously more attractive than static routing but as we saw it earlier, common routing protocols do not support unidirectional links. There are several alternatives to sort this out, according to the network configuration:

1. An unidirectional overlays on top of bidirectional network, as depicted in Figure 2.
2. A mixed network, as depicted in Figure 3.

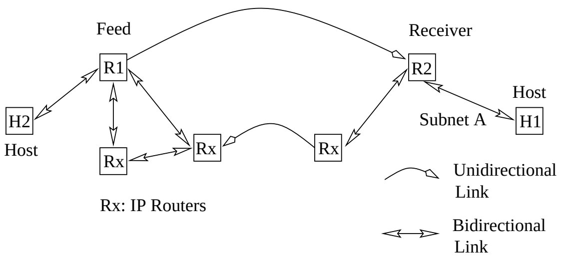

Figure 3: A mixed network

The first case presents a topology that will be quickly deployed in the Internet. The next section (Section 4), explains how to support the unidirectional link in this specific architecture. We describe two approaches, one based on routing protocol modifications and one base on a tunneling mechanism, both transmit routing messages over the bidirectional network. They are considered as short term solutions, they can only handle simple topologies.

The second case correspond to a more general approach. Indeed the topology becomes more complex as unidirectional links are more and more widespread. The short term solution is then not applicable. We are led to design a new protocol which can handle various topologies, it is what we call a long term solution (see Section 5).

4 Short term solutions

We have been interested in problems related to unidirectional overlays on top of bidirectional networks. We proposed a solution based on modifying routing protocols to the first UDLR BoF Session at IETF in June 1996. We also designed a solution based on tunneling. We present hereafter the details of our proposals on both approaches.

4.1 Routing protocol modifications

We were particularly interested in studying two widely used protocols in the Internet: RIP and DVMRP. From their respective RFCs and source code, we proposed some modifications in order to support unidirectional links. The protocol modifications and their experimentations are described.

4.1.1 RIP

RIP is one of the first dynamic routing protocols used in the Internet. It was designed to work in networks where adjacent routers communicate using the same link in both directions.

RIP is based on the exchange of routing information known as distance vectors between directly-connected routers. These distance vectors form a list of destinations a router can reach and at what distance they are (hop count). All routers broadcast routing messages on their interfaces with a TTL of 1 (a routing message is never forwarded). After a certain time lapse, the routers of the autonomous system know all the reachable destinations and there routing tables converge to a stable state.

In Figure 2, because of the unidirectional link receivers are unable to inform feeds of destinations they can reach. They must however send reports to the feeds to indicate the subnets for which they are ready to receive packets.

• **Modifications**
Modifications applied to RIP have been precisely described in [2] and [3]. We give hereafter an overview of these modifications.

Feeds and receivers run a modified version of the protocol. As receivers receive routing messages from feeds, they maintain a list of the potential feeds. This list stores the IP addresses of every feed interface connected to the unidirectional link. In order to substitute the broadcast of routing messages by receivers over the unidirectional link, every receiver sends a copy of the

	R1	R3	R2
ep0	138.96.184.19	138.96.184.1	138.96.184.18
ep1	138.96.184.3	138.96.184.49	138.96.184.50

Table 1: Interface IP addresses

routing messages to the list of potential feeds. These messages are sent periodically to the unicast address of each feed via the back channel.

Routing information sent by feeds should be discarded by receivers, indeed destinations announced by feeds are not reachable through receivers via the unidirectional link. Referring to Figure 2, *R1* informs *R2* it can reach the subnetwork to which *H2* belongs to, at the first announcement, *R2* adds *R1* IP address to the list of potential feeds. Then,*R2* must discard the routing information otherwise it will advertise to the other routers *Rx* that it can reach *H2* via *R1*, which is completely wrong.

Periodically, *R2* sends to the unicast address of *R1* its own routing messages announcing reachability to *Subnet A*. As a feed receives these unicast routing messages, it updates its routing table to take into account the new reachable destinations. Then, it propagates its new distance vectors to the other routers.

• **Experimentation**

Preliminary experiments were conducted at INRIA in order to verify that our proposed modifications work, i.e. that they enable routers to exchange routing messages in the presence of unidirectional links.

The test network is composed of three PCs which have several Ethernet cards, this gives the possibility to set up various topologies. Figure 4 presents the test network that is used to validate the routing daemon modifications.

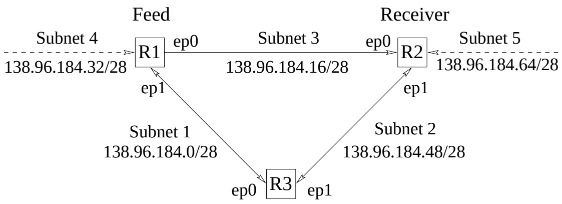

Figure 4: Test network

R1, R2 and *R3* are PCs running the FreeBSD operating system. Subnetworks are set up as depicted in Figure 4, routers IP addresses are summarized in Table 1.

Ethernet links are by nature bidirectional, we have to play around with R2 routing table to prevent it from sending packets on its interface ep0. Deleting the route to *Subnet 3* in *R2* emulates this subnetwork as a unidirectional link. *R1* can forward datagrams over this subnetwork and *R2* can only receive datagrams from it. The other subnetworks are leaved untouched and they simulate the bidirectional network.

In order to dynamically exchange routing information, we need to run a routing daemon on every router. *Gated* is a software program designed to handle dynamic routing with a routing database built from information exchanged by routing protocols. This routing daemon is run on every PC in order to learn and to advertise routes according to the chosen routing protocol specifications (RIP in our case). The code of *Gated* has been modified to support unidirectional links as described in the previous paragraph.

Before running routing daemons over this topology, routers only know destinations corresponding to their directly connected subnetwork. For instance, *R3* knows that it can reach hosts that belong to *Subnet 1* or *Subnet 2*. This is done at bootstrap procedure, when its interfaces are configured with an IP address and a network mask. At this point, if *R3* receives a packet to forward whose destination is *Subnet 5*, it will send back to the source an ICMP message indicating that there is no route to host.

On *R3* we start an unmodified version of *Gated*, this router is not connected to an unidirectional link and it is supposed to behave as any router of the Internet. We then start the modified version of *Gated* on *R1* and *R2*, from this moment on, the routers exchange routing messages and update their routing table. Here is a description of the succession of events which occurs[1]:

- *R2* advertises *Subnet 5* on *Subnet 2*.
- One step latter, *R3* advertises a route to *Subnet 5* on *Subnet 1*. It will be taken into account by *R1* however it is not yet the optimal path.
- *R1* advertises a route to *Subnet 3* on *Subnet 1*.

[1]Not all the events are listed, only the most significant ones

- *R3* advertises a route to *Subnet 3* (via *R1*) on *Subnet 2*.
- Currently, *R1* sends RIP messages on *subnet 3*. The only information that is kept from this message by the routing protocol in *R2* is the message source address (138.96.184.19) that is added to potential list of feeds. From now on, *R2* sends routing messages to the unicast address 138.96.184.19 (*R1*) which is reachable since *R2* learnt from *R3* a route to that destination.
- *R1* receives the routing information from *R2*, it compares the new distance vector to *Subnet 5* to the previous one it had learnt from *R3*. *R1* updates its routing table because the path to *Subnet 5* is shorter directly via *R2* than via *R3*.

At this point *R1* knows that *Subnet 5* is reachable via the unidirectional link. Packets coming from *Subnet 4* and going to *Subnet 5* are forwarded over the unidirectional link. In this experimentation we show that the shortest path including an unidirectional link can be dynamically discovered. The code of the modified *Gated* application is available at [18].

4.1.2 DVMRP

DVMRP is a distance-vector-style routing protocol for routing multicast datagrams through the Internet. It was designed to work on networks where adjacent gateways routing multicast datagrams communicate using the same link in both directions.

Multicast routers exchange routing information using the multicast address 224.0.0.4 on their directly connected subnetworks. The problem here is similar to RIP, receivers cannot send datagrams to feeds.

DVMRP is a protocol described in [16], there is is a draft [1] which explains what modifications could be applied to [16] to support unidirectional links. However, the current implementation of DVMRP, *mrouted*, does not accurately follow the rfc, e.g. DVMRP datagrams carry very different data formats, new features have been added, etc. The only good documentation we found corresponding to the *mrouted* implementation was the C code itself. We therefore propose the following modifications.

DVMRP is based on the calculation of the shortest path tree between a node and the "source" of the multicast datagram. Figure 5 is an example of the ideal path taken by a multicast datagram over a unidirectional link.

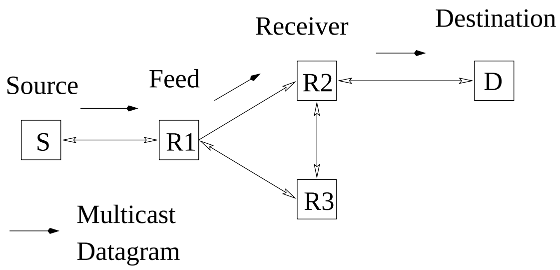

Figure 5: Ideal multicast path (1)

Basically, as a multicast router receives a multicast datagram on one of its interfaces, it forwards it to the others only if the router itself would use this interface to send a datagram to the source of the multicast datagram, i.e. using reverse path forwarding. In other words, if the interface belongs to the shortest path between the router and the source of the multicast datagram, then this datagram can be forwarded.

• Modifications

In the same way as in RIP, when receivers get routing messages on their satellite interface, they maintain a list a potential feeds. Then, every packet which has to be sent over the unidirectional link is copied and sent to the unicast address of each feed. Using this mechanism, DVMRP routing daemon can calculate their routing table "as if" links were bidirectional. However the real topology differs and there are some changes to apply.

Another problem related to unidirectional links, is how can the receiver calculate the shortest path tree between itself and the source of the multicast datagram as depicted in Figure 5? The unidirectional link is a one-way link between the feed and the receiver which is supposed to calculate correctly the reverse path to the source. To hide this physical feature, receivers behave "as if" their virtual interface connected to the unidirectional link can transmit packets. This way, the shortest reverse path tree can be computed by the receivers with the "forward metric" associated to the unidirectional link.

Figure 6 presents another example of ideal multicast packet forwarding in the presence of an unidirectional link.

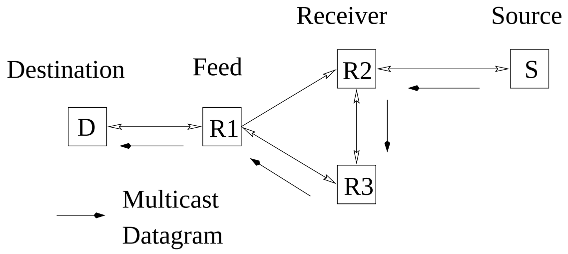

Figure 6: Ideal multicast path (2)

We saw earlier that receivers can send their routing messages to feed via the back channel using the list of IP addresses of the potential feeds. This means a feed can calculate the reverse path tree to the source of a multicast datagram with or without the presence of a unidirectional link. In this example, as the feed receives a multicast datagram forwarded by *R3* whose source is *S*, it checks if the incoming interface belongs to the shortest reverse path tree. For *R1*, the shortest reverse path tree to *S* is via *R2* through the satellite interface, then it discards the incoming multicast datagram. Obviously it is not the desired behavior because the destination will never receive multicast datagrams from the source.

In order to solve this routing problem, as a DVMRP message sent by a receiver arrives at a feed, it must ignore routing information contained in the packet. Only information concerning route reports are ignored, probe, graft and prune messages are still processed to maintain neighboring connectivity. The feed has then no knowledge of possible source beyond the unidirectional link. When the routing daemon calculates the shortest reverse path tree, it will never consider the unidirectional link. Since it does not take into account the path to the source, it chooses the one via the back channel and the multicast datagrams are then routed according to reverse path forwarding.

• **Experimentation**
We use the same topology as in Figure 4, *R1* and *R2* run a modified version of *mrouted* and *R3* runs an unmodified version.

We did the following scenario to validate the modifications: a host connected to *Subnet 4* generates multicast traffic and a host connected to *Subnet 5* joins the corresponding multicast group. As expected the multicast traffic was going over the unidirectional link. The second experiment was to have the emitting host on *Subnet 5* and the host joining the multicast group on *Subnet 4*. The traffic was normally routed over the bidirectional links.

4.2 Tunneling approach

Tunneling is a mean to construct virtual networks by encapsulating the data. This generally allows new protocols experimentations and provides a quick simple solution to various routing problems by building a kind of virtual network on top of the physical one. The aim of using tunneling is to make routing protocols work over unidirectional links without having to bring any modification to them.

The tunneling approach adds a layer between the network interface and the routing software on both feed and receiver sides (or between some intermediate routers), resulting in the emulation of a bidirectional satellite link where only an unidirectional link is available. Packet encapsulation is hiding the actual topology of the network in order to elude the routing protocols and make them behave as if there exists a bidirectional link.

4.2.1 Tunneling mechanism

Basically, a tunnel is established between a receiver and a feed. Routing traffic is sent from the receivers to the feed through the tunnel, i.e. captured by the added layer interface which encapsulates it and sends it on the actual reverse link. The added layer in the feed station decapsulates it and delivers it to the routing protocols as if this was coming from the unidirectional link.

Figure 7 shows the set up of a tunnel between a feed and a receiver. The *ef* (respectively *er*) interfaces correspond to the feed (respectively *receiver*) interfaces, the index *0* (respectively *1*) stands for an unidirectional (respectively bidirectional) interface.

As *R2* needs to send a routing message to *R1* (the destination address is either *ef0* or the broadcast address of *Subnet 3*), it is tunneled via the interface *er1*. The packet is encapsulated in an IP packet whose IP source address is *er1* and IP destination address is *ef1*. The datagram is then sent to the end point of the tunnel via the back channel. As it is received by the feed, the pay-

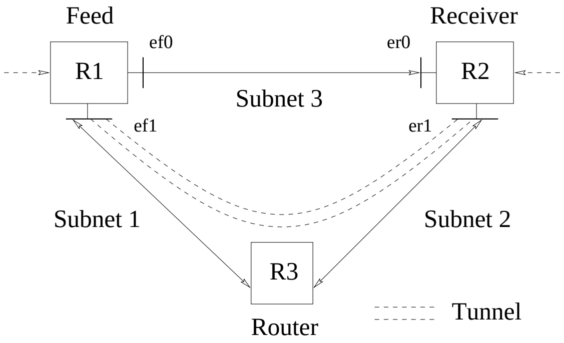

Figure 7: Tunneling between a feed and a receiver

load of the datagram is decapsulated. The new IP packet that is obtained is routed according it its destination address which is the IP address of the interface *ef0* or the broadcast address of *Subnet 3*. The packet is then delivered "locally" and not forwarded since the destination address is the feed itself. The IP stack passes the packet to higher level, in our case the routing protocol.

As in the routing protocol modification approach, we need a mechanism to dynamically discover feeds and set up tunnels. When a receiver boots up it must discover the presence of feeds dynamically and creates a tunnel with each of them. The tunnel end points are on the bidirectional network (see Figure 7). The feed sends its routing messages on *Subnet 3* using *ef0* IP address as a source address. The receiver cannot take this source address as end point of the tunnel otherwise a tunnel loop occurs. The only way a receiver can learn the tunnel end point (i.e. a feed interface belonging to the bidirectional network) is to define a new simple protocol. Feeds periodically advertise their tunnel end point (*ef1* IP address) over the unidirectional link. As a receiver gets this message, it checks if the tunnel exists, if not it creates a tunnel and uses it.

Routing protocols usually have mechanisms based on timeouts to detect if a directly connected network is down. If the unidirectional link is not operational and receivers keep on sending there routings messages via the back channel, feeds will continue to send packets on its unidirectional interface. This is an undesirable behavior because we create here a black-hole (see Section 3.1). In order to prevent this, if receivers do not get routing messages for a certain a time lapse from the unidirectional link, they turn their tunnel interface off. As a result, feeds receiving no routing messages from receivers, delete all destinations in their routing table which are reachable via the unidirectional link.

Having a tunnel between a receiver and a feed is very attractive because the unidirectional link is totally hidden to applications. As far as routing protocols are concerned they have to be correctly configured. The two routing protocols we saw in Section 4 can be configured as follow:

- with RIP, receivers must not take into account destinations advertised by feeds (see Section 4.1.1). This can be done by setting the metric of the receiver unidirectional interface to infinity. All distance vectors coming from this interface have infinite value and routes are then ignored.
- with DVMRP, the metric of the feed unidirectional interface is set to infinity. The shortest reverse path to a source from a feed is then never through the satellite link. As a result multicast packets can be forwarded over the satellite link. In our experimentation, receivers have two interfaces: a unidirectional interface and an Ethernet interface connected to a local network. Receivers set the infinite metric to its Ethernet interface. As a multicast packet arrives on the Ethernet interface, the reverse path to the source has an infinite value and the packet is not tunneled. This prevents receivers from sending multicast traffic in their tunnel(s) to the feed(s). Newest multicast routing daemons allow filtering according to destinations addresses, this is also an alternative to prevent multicast traffic from being tunneled.

Multicast routing with satellite links is being experimented over Europe. The up-link station is located at INRIA Sophia-Antipolis (France), receivers at University College London (England), Forschungszentrum Informationstechnik GmbH (Darmstadt, Germany) and Eutelsat (Paris, France). Multicast routers are being configured to get the optimal routing. As part of the experimentation, SIGCOMM 97 which is held in Cannes (France) in September, will have several sessions retransmitted on the Mbone via the satellite link.

We described how routing protocols can work over the tunneling mechanism, we now have to define the type of encapsulation to transport IP packets.

4.2.2 What encapsulation?

Several encapsulations are possible such as IP over IP [12] and Generic Routing Encapsulation [6, 7]. The choice of the encapsulation depends on the underlying address resolution protocol.

The Address Resolution Protocol (ARP) [13] is a "method of Converting Protocol Addresses (e.g, IP addresses) to Local Network Addresses (e.g., Ethernet addresses)". This protocol assumes that links are bidirectional, if the link is unidirectional, a receiver cannot respond to an ARP request generated by a feed.

There are several propositions to figure out how to convert an IP address to a hardware address:

1. Hardware addresses are derived according to a one to one mapping function from IP addresses. To one IP address corresponds a unique hardware address that is derived from it. Knowing the function, a feed can obtain the hardware address of a receiver without using ARP like protocols. No ARP mechanism is required.

2. Hardware addresses are statically maintained in feeds. A similar solution is adopted in cable modems [17]. No ARP mechanism is required.

3. A feed sends an ARP request over the satellite link, only one receiver can issue a response, it is the receiver whose IP address has to be converted. Its ARP response must be tunneled to be delivered to the feed, the end point of the tunnel. GRE supports the encapsulation of ARP messages and IP packets (for routing information). A GRE packet is composed of a header in which a type field specifies the protocol that is encapsulated (ARP, IP, IPX, etc.). In our case only the support of ARP and IP should be implemented.

   A packet tunneled with this encapsulation has the following format: the delivery header is a IP header (IP is the delivery protocol) whose destination address is the tunnel end point, it is followed by a GRE header which indicates that the payload is either an ARP message or an IP packet. Table 2 presents the encapsulation.

IP hd	GRE hd type = ARP|IP	payload = ARP|IP

   Table 2: Format of tunneled packets

   As a feed receives this packet, it checks the type field in the GRE header, if the payload is an IP packet, it is routed (See Section 4.2.1), if the payload is an ARP response, it updates its ARP table.

4. Receivers can send periodically to the feeds their hardware address together with the corresponding IP address. Feeds update their ARP table without the need of an "on demand" mechanism. These periodical messages are not necessary sent through the tunnel and then there is no need to use GRE encapsulation. Indeed, since the tunnel is only used to transmit IP packet, a simple encapsulation such as IP over IP is enough. Table 3 presents the encapsulation.

IP hd type = IP>IP	payload = IP datagram

   Table 3: Ip over IP encapsulation

   As a feed receives this packet, it decapsulates the IP packet and routes it (See Section 4.2.1).

Both types of encapsulation are currently discussed in the UDLR Working Group. They depend on the ARP mechanism that will be used.

Other tunneling mechanisms are proposed by the WIDE group [8], the Aerospace corporation [15] and Hughes Research Laboratories [19]. Details of the different mechanisms are discussed within the UDLR Working Group.

4.3 Protocol Modification or Tunneling?

Routing protocol modification and tunneling are two different approaches. They are however similar in many aspects. We compare hereafter their suitability to provide an efficient solution for the short term case. As described in detail in the two previous subsections, both approaches indeed solve the main problem: how receivers send routing information back to feeds. However, several other issues such as dynamic configuration, efficiency, multicast support, scalability and ease of deployment need to be addressed in order to complete the comparison.

Dynamic configuration Two issues are considered, (1) metric selection for the reverse link in the case of tunneling and (2) dynamic configuration of tunnels. In the tunneling approach the metric of the virtual reverse link should be selected in such a way that return data traffic from receivers to feeds is not encapsulated[2]. A value of 15 may be selected for RIP. The same technique may be adopted for DVMRP. As for OSPF, the metric should be higher than the distance between the receiver and the corresponding feed. It is therefore easier to perform these configurations dynamically in the case of routing protocol modifications as the information on the topology is available to the routing protocol. As for the feeds and receivers dynamic detection, it is implicit in the routing protocol modification approach, while it requires a specific protocol in the tunneling approach as described in [8] and [15].

Efficiency In one case the modified protocol routes the packets without adding any encapsulation overhead, while in the second case, packets are encapsulated by the tunneling mechanism at the link level. However the encapsulation overhead added by the tunneling mechanism is not an important issue because it involves a few byte headers only for routing traffic sent on the back channel.

Multicast routing support The routing protocol modification indeed allows for the support of multicast datagram forwarding over the unidirectional link. However, the tunneling mechanism may also be used to support DVMRP if properly configured as it was demonstrated by the WIDE project.

Scalability Both approaches face the same scalability problems: how they will behave in the case of a large number of receivers and/or feeds. Multiple feeds are possible in services like "Skyplex". Again, the use of multicast routing on the bidirectional network would help to solve this open issue. In the case where a receiver would like to communicate with multiple feeds, both approaches allow to use multicast on the bidirectional network either by having the feeds subscribing to a feed multicast group in the first case or by supporting a multicast tunneling mechanism in the second case [10].

[2] In some cases, the use of the back channel to carry upper level protocols data may be desirable. It is not difficult to enable this in the tunneling approach.

Inter-domain routing Because the footprint of the satellite is by nature very large, receivers are likely located in different Autonomous Systems (AS). None of the approaches has considered the problem of inter-domain routing. This has to be dealt by Border Gateway Protocol (BGP) [14]. Dealing with unidirectional links might not bring any trouble since asymmetric metrics can easily be set administratively. But, unlike current routing protocols, BGP runs over TCP, which might not scale well since the feed should keep state about each connection. Moreover, an important issue is where running BGP. At receivers only, receivers becoming part of the feed AS; at feeds only; or both at feeds and receivers. This is detailed in [4].

Ease of deployment The "short term" solution is needed *now*. Several satellite operators and service providers would like to offer Internet access via satellite. The tunneling solution is undoubtedly favored on this aspect. In fact, deploying modified protocols may take time (for OSPF all routers should support the modified implementation, while in RIP and DVMRP only feeds and receivers need to do so). On the contrary, installing the tunneling mechanism allow transparent operation for all upper level protocols[3].

The tunneling approach seems therefore more appropriate to be the "short term" solution. In fact, the UDLR working group decided to focus on tunneling in his last meeting held at Memphis.

5 Long term solution

We are foreseen an increasing need for unidirectional links. There are situations where it may

[3] Transparent does not imply optimal operation.

not be possible to set up bidirectional links between two nodes, for some cost or technical reasons. For instance, GEO satellites with receive-only hardware, ad-hoc (wireless) networks where pairs of routers do not have the same transmitting power one to the other. Moreover, some existing devices used for some other type of data transfer, such as TV on cable, are sometimes unidirectional and do offer spare bandwidth. Their providers are willing to offer connectivity to the Internet via this kind of device as satellite providers do.

Basically, solutions for the short term are not applicable for those configurations where many links could be unidirectional, i.e. in mixed networks.

Tunneling works fine as soon as there is only one unidirectional link on top of a bidirectional network, but it will not work if the path between the receiver and the feed comprises itself an unidirectional link, and therefore requires the setting of a tunnel, this leading to deadlock, as shown on figure 3.

It is neither not possible to consider a solution based on modifying existing routing protocols because mixed networks have specific characteristics that have impact on routing techniques currently in use, namely Link State (LS) and Distance Vector (DV) as we will see in 5.1.

It therefore emerges that we should rather design a targeted solution, specifically adapted, that explicitly and optimally takes into account the underlying architecture of the network. It seems far better to treat the problem from scratch since DV and LS are not directly applicable. We therefore focus on unidirectional link networks since this allows to address the most general case, although this might not provide the most optimal solution in mixed networks.

Another reason why we focus on unidirectional link networks is that existing routing protocols assume that pairs of routers are connected, not only by bidirectional links, but by symmetric bidirectional links. Traffic is by nature asymmetric, especially web traffic, but existing routing protocols don't allow to consider distinct bandwidth in each of the two directions. Splitting up one bidirectional link in two unidirectional links allows to solve at the same time the question of asymmetric routing since this allows to set distinct metrics on each unidirectional link.

Our study shows that a new technique, based on circuit detection, is well adapted to the specific characteristics of unidirectional links as described in section 5.2. Section 5.3 gives an overview of our protocol. A more detailed description is given in [5].

5.1 Current techniques are not applicable

Link State In LS protocols, flooding is used to disseminate to the entire network each node connectivity, i.e. links state. This flooding of LS requires hop-by-hop acknowledgment in order to ensure the consistency of the information contained in routing tables at each node. Hop-by-hop acknowledgment is obviously not possible in mixed networks. First, acknowledgments would likely travel through a longer path, this leading to overhead. And second, it is not even possible to send acknowledgments at the time of the topology discovery since nodes don't know how to forward them to a particular destination.

Distance Vectors In DV protocols, each node broadcasts to its neighbors vectors of distances containing the shortest distances to all destinations it knows. Each node concludes it can itself reach the destinations via the node that advertises it, with a distance increased by the distance from the neighbor to it. Nodes therefore assume that paths are bidirectional and costs symmetric, which is not longer true in a mixed network. A node cannot determine the destinations it can reach and neither the distances using this techniques. DV can only provide the knowledge of nodes in the network without providing routes to get to them.

5.2 A circuit-based technique

We need a new technique that doesn't require the sending of hop-by-hop acknowledgments and that doesn't assume that links bidirectionality neither paths and distances symmetry. The communication still need to be bidirectional, since no communication is possible if one-way connectivity only is ensured. Determining a path that gets to a particular destination is therefore not sufficient. We also need to determine a path that comes back.

The characteristics depicted by unidirectional links favors a new technique based on the detection of circuits, i.e. a path that goes to a destination and goes back to the departure place.

Possible destinations in the network are therefore all nodes that appears on circuits, what we call circuit members. Determining a circuit therefore provides the knowledge of possible destinations in the network, a route to get to them, and ensure their reachability in both directions.

Correct routing is guaranteed if all members of a circuit share the same knowledge. In this case, there is no need for hop-by-hop acknowledgment if each node knows that the precedent node has an up-to-date view of the circuit. Routing of data packets can therefore start straight on circuits that have been found even if the full network topology is not known yet.

The cost to join a destination of the circuit is the same for all members of the circuit and is equal to the cost of the outward path from the departure place to the destination plus the cost of the return path, as opposed to twice the metric in a bidirectional network.

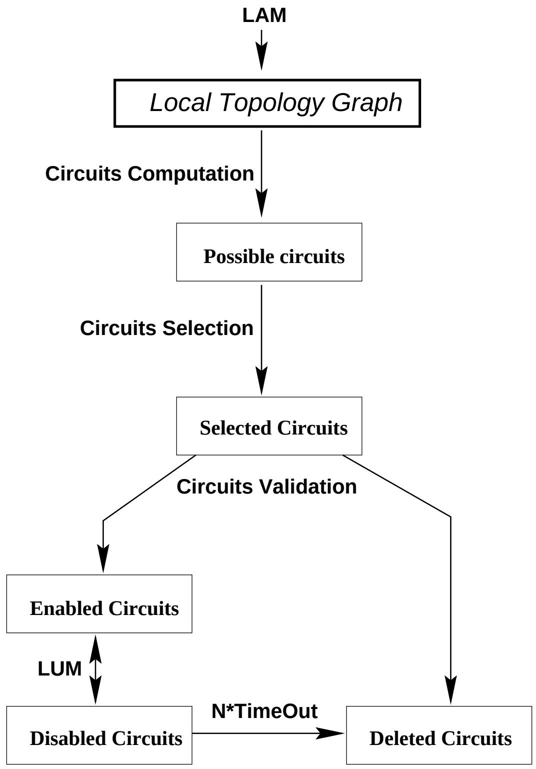

Figure 8: Circuit selection and validation

5.3 A Circuit Discovery Unidirectional Link Routing Protocol

The protocol allows dynamic routing in unidirectional networks. It is based on the detection of circuits. It computes circuits at each node from a local topology graph representing the network.

The topology graph is obtained by *Link Advertisement Messages* (LAMs) containing information describing new links. Upon reception of a LAM containing information not already recorded in the local topology graph, a node updates its local topology graph and sends a LAM containing only the new information on all outgoing links, i.e. to all its successors.

Circuits are computed from the topology graph and are labeled “possible”. A decision process selects the best amongst them according to various metrics. Selected circuits are thereafter validated by a *circuit validation message* (CVM). The CVM starts at the node on the circuit with the smallest IP address, called the *circuit origin* and is forwarded to the next node appearing on the circuit. If the validation procedure succeeds, the circuit is labeled as “enabled”. The routing table is then based on the information contained on those circuits.

The identity and operationality of predecessors is obtained by periodic *KeepAlive Messages* (KAMs) sent to all successors. Their non-delivery informs successors of a failure occuring on an incoming link or at the predecessor itself whereas their resumption means the link or node has recovered from the failure.

Successors have the responsibility to monitor KAMs and to announce new links in a LAM and link updates i.e. link failures and/or recoveries in a *Link Update Message* (LUM). LUMs are only forwarded to nodes appearing on circuits involved by the update. Upon reception of LUMs, a node marks relevant links in the topology graph and relevant circuits in the table of circuits as “disabled” in the case of failure or “enabled” in the case of recovery. Disabled circuits and links are definitely deleted after a specified amount of time.

In short, LAMs brings the knowledge of the topology while LUMS ensure operationality of links and circuits. By default, nodes consider their predecessors as operational only if they receive KAMS, while they consider their successors as operational as long as they don't receive LUMs indicating their failure. Circuit consistency is ensured by the validation of the selected circuits, and is maintained by the LUMs.

We have simulated the behavior of our protocol on a test network and it works fine.

6 Conclusion

In this paper, we have presented a general description of our work on dynamic routing in Networks with Unidirectional links. Tunneling is preferred to routing protocol modification as a short term solution for the case of unidirectional overlays on top of bidirectional network. This solution is being finalized within the IETF UDLR Working Group. We are currently implementing and experimenting this solution over a test network at INRIA. The next step is to have European wide experiments using a Eutelsat satellite and involving partners from the MERCI project. We also showed that a new protocol is needed to address the routing problem on mixed networks. We described a protocol that provides dynamic routing in a unidirectional network. Future work is to simulate the protocol behavior on different network topologies and to study how to perform efficient routing in the case of mixed networks.

References

[1] Emmanuel Duros and Walid Dabbous. Handling of Unidirectional Links with DVMRP. Internet draft <draft-ieft-dvmrp-unidirectional-link-00.txt>, INRIA Sophia-Antipolis, March 1996. available at http://www.inria.fr/rodeo/udlr.

[2] Emmanuel Duros and Walid Dabbous. Supporting Unidirectional Links in the Internet. In *Proceedings of the 1st International workshop on Satellite-based Services*, Rye, New-York, November 1996. INRIA Sophia-Antipolis, France.

[3] Emmanuel Duros and Christian Huitema. Handling of Unidirectional Links with RIP. Internet draft <draft-ieft-rip-unidirectional-link-00.txt>, INRIA Sophia-Antipolis, March 1996. available at http://www.inria.fr/rodeo/udlr.

[4] Thierry Ernst. Dynamic Routing in Networks with Unidirectional Links. Master's thesis, Université de Nice-Sophia-Antipolis, June 1997.

[5] Thierry Ernst and Walid Dabbous. A Circuit-based Approach for Routing in Unidirectional Link Networks. INRIA Sophia-Antipolis, France, 1997. Submitted to INFOCOM'98.

[6] S. Hanks, T. Li, D. Farinacci, and P. Traina. Generic Routing Encapsulation. Request for Comments 1701, NetSmiths Ltd and Cisco Systems, October 1994.

[7] S. Hanks, T. Li, D. Farinacci, and P. Traina. Generic Routing Encapsulation over IPv4 networks. Request for Comments 1702, NetSmiths Ltd and Cisco Systems, October 1994.

[8] Hidetaka Izumiyama and Akihiro Tosaka. Unidirectional Link Routing with IP tunneling. Internet Draft <draft-ietf-wide-udlr-vif-00.txt>, Keio University, WIDE Project, April 1997. available at ftp://zenon.inria.fr/rodeo/udlr/archive.txt.

[9] G. Malkin. RIP version 2 - Carrying Additional Information. Request For Comments 1723, Xylogics, Inc, November 1994.

[10] D. Meyer. Representing Tunnels in RPSL. Internet draft <draft-ietf-rps-tunnels-00.txt>, University of Oregon, November 1996.

[11] J. Moy. Open Shortest Path First - OSPF version 2. Request For Comments 1583, Proteon, Inc, March 1994.

[12] C. Perkins. IP encapsulation within IP. Request for Comments 2003, IBM, October 1996.

[13] David C. Plummer. An Ethernet Address Resolution Protocol. Request For Comments 826, MIT, November 1982.

[14] Y. Rekhter and T. Li. A Border Gateway Protocol 4 - (BGP-4). Request For Comments 1771, T.J. Watson Research Centre, IBM Corp - Cisco Systems, March 1995.

[15] James Stepanek and Stephen Schwab. Virtual internet Packet Relay - An Encapsulation Architecture for Unidirectional Link. Internet draft <draft-stepanek-vipre-00.txt>, The Aerospace Corporation and Twin Sun Inc, March 1997. available at ftp://zenon.inria.fr/rodeo/udlr/archive.txt.

[16] D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. Request For Comments 1075, BBN STC & Stanford University, November 1988.

[17] Cable Modem web page. http://www.cablemodem.com.

[18] UDLR web page. http://www.inria.fr/rodeo/udlr.

[19] Yongguang Zhang webpage. http://www.wins.hrl.com/people/ygz.