DYNAMIC ROUTING IN NETWORKS WITH UNIDIRECTIONAL LINKS

A dissertation submitted for the
MASTER OF SCIENCE
"NETWORKS & DISTRIBUTED SYSTEMS"
University of NICE-SOPHIA-ANTIPOLIS - FRANCE

Work done at INRIA Sophia-Antipolis
project RODEO

Thierry Ernst supervised by Walid Dabbous

20/06/1997

Acknowledgements

I would particularly like to thank Walid Dabbous, my supervisor, for all his accurate comments and feedback during the placement and while writing this report. I'm also grateful grateful to the Rodeo team and the Inria which provided me the best possible technical and human environment to make this placement a very enriching experience.

Contents

Abstract

Satellites offer a large bandwidth capacity and a large geographical coverage. Their broadcast nature could be very useful for multi-casting purposes. They can as well bypass bottlenecks by providing high bandwidth link over large geographical areas.

Those advantages are of great interest to the Internet community but the cost of a feed is too prohibitive. Only a few big organizations or services providers could afford it. There is nevertheless a cheap solution to provide one-way connectivity to the Internet using satellites, but on the condition that there exists a terrestrial back-channel since two peers always need to establish a communication in both directions. The idea is to add an alternative satellite unidirectional link on top of a bidirectional network in order to improve the connectivity.

The problem comes at the level of routing since existing protocols cannot be aware of the existence of networks reachable through the satellite unidirectional link. The back-channel could be used to transmit the necessary routing traffic, but routing protocols automatically discard routing responses coming through another link that the one requests were sending to.

There are two distinct approaches addressing this problem. Tunneling, and routing protocol modification. Tunneling is a hack, but it allows to give a quick solution without requiring any modification in existing routing protocols, and neither in upper layers protocols and softwares. Routing protocols modification is more elegant, it explicitly deals with the problem, but it requires changes in routing protocols implementation and may be in some other upper layers protocols which are highly dependent of the underlying structure. Tunneling is therefore favored for the short term.

As we need to take a look to the long term, we are likely foreseeing a widespread use of unidirectional links. We could have unidirectional links connecting set of networks, and we could have networks where all links appear unidirectional. In those conditions, we need to give optimum solutions to the use of unidirectional links. Tunneling is obviously not a valid long term solution.

We will therefore consider the applicability of current protocols techniques to support unidirectional link routing: Link State and Distance Vectors. There are serious limitations since most protocols assume that packets travel on the same path in both directions whereas our aim is to allow paths to be asymmetric, i.e. the reverse path is taking a different route from the outward one. The Link State technique has limitations because it requires hop-by-hop acknowledgment of routing traffic while Distance Vectors cannot provide the cost to get to the others nodes without important modifications.

Since unidirectional links exhibit some specific characteristic, a new technique specific to UDLs may provide an alternative to Link State and Distance Vectors. We propose such solution detecting circuits which are paths originated at the local node that go back to the local node likely on a different path. Nodes collected on those paths are accessible nodes. We called our prototype protocol “Circuit Discovery UniDirectional Link Routing Protocol”.

Solving unidirectional routing as well solve the question of asymmetric links since a bidirectional link is split off in two unidirectional links.

Introduction

The Internet is a widespread and heterogeneous network of computers interconnected by various means. Routers are those computers connected to more that one network. They are the entities responsible for the accurate distribution of the traffic throughout the Internet. Until now it was assumed that pairs of routers were connected by bidirectional and symmetric links.

In practice, links are not really symmetric. Existing links are bidirectional, but there could be a distinct bandwidth capacity in each of the two directions. This asymmetry is not taken into account for routing decisions. Basically, non optimal decisions are made because the actual topology of the network is not fully exploited.

Moreover, unidirectional links (UDLs), i.e. links with zero return bandwidth, are emerging. They have first appeared in the shape of satellite links which exhibit some interesting characteristics, like their broadcasting nature and their large geographical coverage.

Actually, there already exist satellite bidirectional links in the Internet. The cost of feeds, the devices allowing emission of data toward satellites, is nevertheless too important to generalize their use. However, the cost is negligible for receive-only hardware. Satellite operators are therefore willing to provide one-way, i.e. receive-only connectivity to the Internet via satellite UDLs. This would serve numerous subnetworks attached to reception-only satellite dishes.

Since we could foreseen an increasing number of satellite links, new kinds of networks are likely going to appear soon or later. We therefore need to consider the problem of routing in a network where UDLs are becoming numerous on two levels: inter-domain routing and intra-domain routing. Inter-domain routing involves to interconnect autonomous systems (ASs) via UDLs, whereas intra-domain routing involves routing in networks where all links could be UDLs.

Before this study, UDLs were seen by the Internet community as bidirectional links with a higher metric in the reverse direction. UDLs were therefore seen as a special case of asymmetric bidirectional link (BDL). The solution to routing over UDLs was then to allow routing protocols to take metric links asymmetry into account. In asymmetric links, there is likely to exist enough return bandwidth for the routing traffic whereas there is no return bandwidth at all if the link is unidirectional. The problem look easier if we transpose the way around to say that asymmetric links are in fact two UDLs with different metrics. A network entirely made of UDLs is therefore the general case and is referred by the term "All-UDL".

Resolving the All-UDL case therefore solves at the same time the problem of asymmetric routing since an asymmetric BDL is represented by two UDLs.

In chapter 1 we present the different cases where UDLs could apply. In the first one, an alternative satellite UDL on top of a BDL network provides a better connectivity to the Internet. In the second, we consider BDL networks linked by UDLs, and in the third, the general case where all links are UDLs.

Chapter 2 considers the first case. Various proposals divided into two distinct approaches are presented there, discussed and compared. One approach, tunneling, is favored in addressing the problem for the short term while the other, routing protocol modification (RPM) is better for the long term. Problems still to resolve are much the same for both approaches. Those basically are scalability and inter-domain routing. A specific section shows the limitations of the solutions when inter-domain routing is involved and we give an outline of possible solutions.

Chapter 3 considers the others cases of UDLs which are seen as long term applications and could be solved by a general solution. It emerges that a special purpose protocol exploiting specific characteristics of UDLs. We study routing techniques used by protocols the most currently in use in the Internet, namely Link State (LS), Distance Vectors (DV), and in some extent Path Vectors (PV) and we show in which respect it could apply to the general case of UDLs. None of them is suitable without modifications. We believe that a new approach would give a more optimal solution.

In chapter 4, we present our proposed unidirectional link routing protocol. The prototype is called "Circuit Discovery Unidirectional Link Routing Protocol" (Circuit Discovery UDLRP) and is based on the detection of circuits.

Open issues that are left out for further studies are presented in chapter 5. We are currently working on it.

Finally, chapter 6 concludes with this report.

Chapter 1

The different cases of Unidirectional Links

Various situations where UDLs could apply have been identified so far. Those are detailed in the following sections. In the first situation an alternative satellite UDL is set on top of a bidirectional network in order to bypass bottlenecks and provide a better connectivity to the Internet. A feed is sending out data to a satellite that forwards it to receiver-only terrestrial stations. In the second situation, *bidirectional islands* within which all links are bidirectional are *interconnected by unidirectional links*. In the last one, we consider a network where *all links unidirectional*.

1.1 An Alternative UDL on top of a BDL network

Research considering UDLs has originated with this case and has been initiated by satellite operators who are willing to carry out the Internet's traffic on satellites in order to make those more profitable since they can offer high bandwidth over a large footprint. On the other hand, the Internet community sees many interests in satellites links. Those could be used in various situations.

First, the broadcasting nature of satellites allows to send data simultaneously to many receivers. An important amount of bandwidth could therefore be saved using satellites when numerous receivers are concerned. Broadcasting data, multi-casting involving many and widespread receivers, interconnecting networks located a long way away from one another are seen as possible uses.

Second, satellites offer an easy connectivity to the Internet for mobile networks.

Third, there are cases where satellites could be used to improve the connectivity to the Internet. It could bypass a bottleneck by providing a high bandwidth link on large geographic regions.

If many customers would in theory be interested by satellite connectivity to the Internet, this requires an important money investment a few of them are able to pay since the feeds, the devices allowing emission, are very expensive. On the other hand, the investment allowing reception of data via the satellite is negligible since reception antennas are cheap. Moreover, most of the traffic in the Internet is asymmetric in the sense that there is more traffic flowing from the service provider to the receivers than in the opposite direction. Traffic originated at the receivers is essentially made of requests of services and acknowledgments of data.

Satellites could therefore be used to provide a semi-connectivity to the Internet, but a back-channel is required since most of the protocols and software running on the Internet at least require a mean to acknowledge proper reception of data. Figure 1.1 shows this architecture.

In this architecture a bilateral communication is ensured between a service provider and its receivers. The outgoing connectivity (i.e. from the sender to its receivers) is offered by a satellite UDL. Basically, a feed sends data to the satellite which broadcasts it to the receivers that belong to the satellite footprint. Receivers have the ability to receive data from the satellite and send it back to the feed via a back-channel, like for instance a PPP dialup line or a cellular modem in the worst case, or a proper Internet

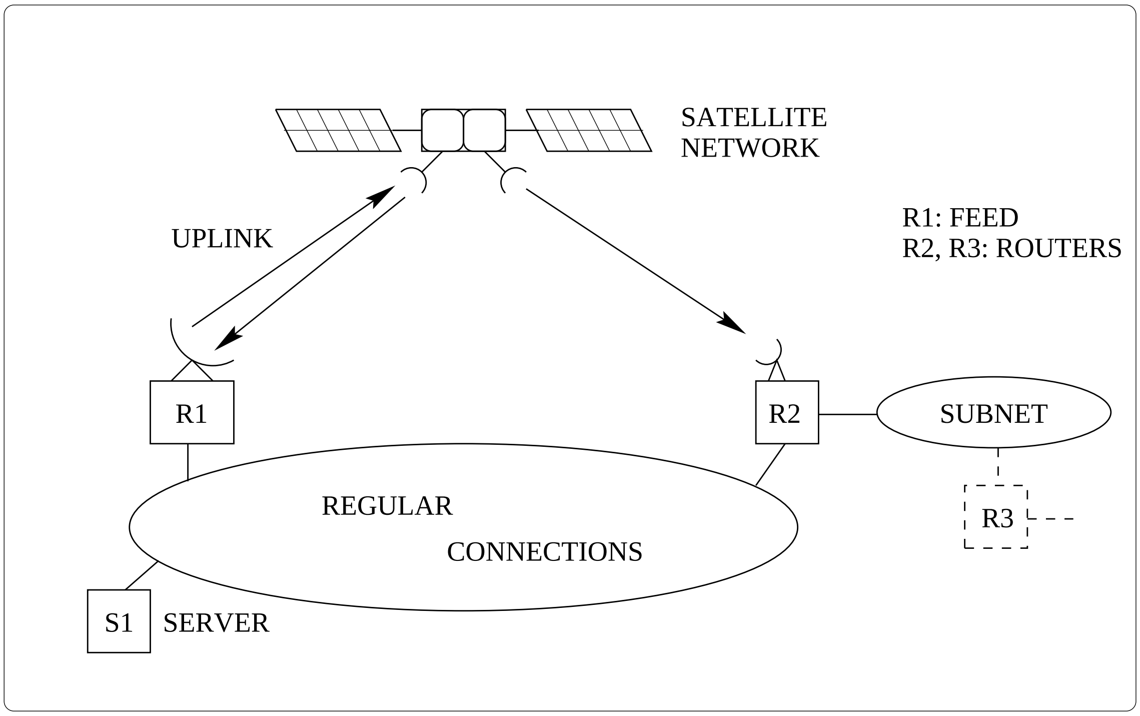


Figure 1.1: Satellite unidirectional link

connectivity. This terrestrial connectivity is bidirectional, but, because its bandwidth is very low or very expensive, it is only used to send requests and acknowledgments from the receivers.

This system is described in [2] an makes an asymmetric bidirectional communication out of an satellite UDL.

In the context of our study we only focus on routing. Routing is not involved if receivers are single hosts, but we want to grant access to subnetworks that could be joined through the satellite UDL. We are interested in having the knowledge of subnetworks reachable through the satellite in order to route traffic through it.

UDLs may apply to others types of links such like cable that present properties similar to those of satellite UDLs.

1.2 Bidirectional Islands Interconnected by Unidirectional Links

In our context, islands are entities inside which all nodes can establish a bidirectional communication with any other node without the help of any external link. The only requirement is that everyone is able to join everyone. A graph representing such an entity is therefore connex. Generally speaking, everything that happens inside islands should be transparent from the overall picture of interconnected islands as soon as a local path is not routed between islands. What happens inside doesn't apply in the present section and should be dealt later on. We could basically have an UDL in an island as shown on figure 1.2 where there is an UDL in island 4 from A to B and C. The graph representing the island remains connex since B and C can establish a reverse path inside the island without requiring to route through another island. If that was necessary, we should partition this island in two different islands.

The term “islands” emphasizes that islands are isolated from the remainder of the Internet. They are isolated in the sense that there are no standard BDL links binding one island to another. Basically, they are linked by UDLs.

The architecture depicted here is therefore a higher level network made of UDLs that interconnects islands. Two islands could be interconnected by several of those UDLs, but it is likely that there is no

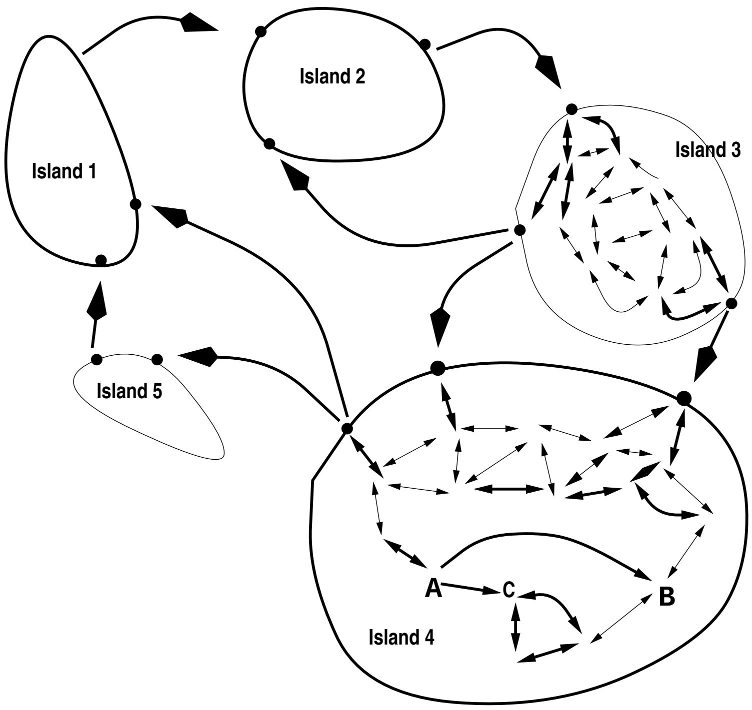


Figure 1.2: BDLs Islands Interconnected by UDLs

link in one of the two directions. Traffic therefore need to be relayed through some third party islands.

This architecture could for instance have an interest for inter-domain routing if we consider that islands are in fact ASs.

1.3 All links are UDLs

This is the general case where all links are unidirectional within a subset, i.e. an AS of the Internet, or even in the entire Internet. Bidirectional links between two peers could still exist, but those may be split in two unidirectional links for the sake of simplicity. Current routing protocols assume that links are bidirectional and symmetric, but there is often a distinct bandwidth in each of the two directions. This is not taken into account but could be very useful when quality of service routing is required. This justify the consideration of this particular architecture where all links are UDLs, since solving this general case addresses the question of asymmetric links at the same time. Splitting one BDL becomes two different links allows to set easily different metrics.

This architecture could also be of great interest in ad-hoc wireless networks where pairs of routers do not have the same transmitting power one to the other.Another area of interest is packet radio networks due to the mobility of networks. Unidirectional routing could for instance enhance the location of the home agent since packets are rerouted in order to reach a mobile computer or network.

Chapter 2

Alternative UDL on top of a BDL network

In order to use this architecture, we need to solve two problems: how feeds and receivers detect each other dynamically and how receivers send routing information to feed since there is no reverse satellite link.

Problems arise because existing routing protocols in the Internet assume that links are bidirectional. At the feed, the routing protocol expects that routing messages advertising subnetworks beyond the UDL are coming via the same link which is impossible. The terrestrial return path could be used to transmit this information, but routing protocols automatically discard routing data coming through another interface that the one it was sending to. Those networks beyond the satellite could effectively appear reachable via the normal terrestrial link, but there are going to look unreachable via the satellite link. No data would be sent through it.

Because satellites are able to broadcast data to a huge set of customers, solutions based on static routing where routes are manually set up to elude routing protocols are not acceptable. Dynamic routing is therefore required.

A specific working group called "Unidirectional Link Routing" (UDLR) at the IETF currently addresses the problem of UDLs over satellites in the short term. In the next sections, we present and discuss the distinct propositions debated in this working group. The proposals are basically scattered into two approaches, tunneling and routing protocol modification (RPM) which both solve the above problems. For more details about outgoing work in this working group, we recommend the reader to have a look at the UDLR's web page [29] and in the archives of the mailing list [1].

In the next sections, we have exhibited some of the characteristics of both approaches and we have compared them in order to study their applicability. This comparison provided a base discussion and helped in enhancing the discussion within the working group which finally agreed during the last meeting that tunneling should be favored in order to give an immediate solution. Our discussion showed that the problems to resolve where much the same whether tunneling or RPM is considered. In those conditions, modifying routing protocols would be to cumbersome.

Although tunneling is the UDLR's preferred solution in the short term, RPM is turning out to be more promising considering the long term, because it explicitly takes into account the actual underlying structure of the network.

Nonetheless, existing solutions have not considered the questions of inter-autonomous system routing. We fill this gap and show how importance this point is, together with the problems that this could cause and various proposals of our own that could apply.

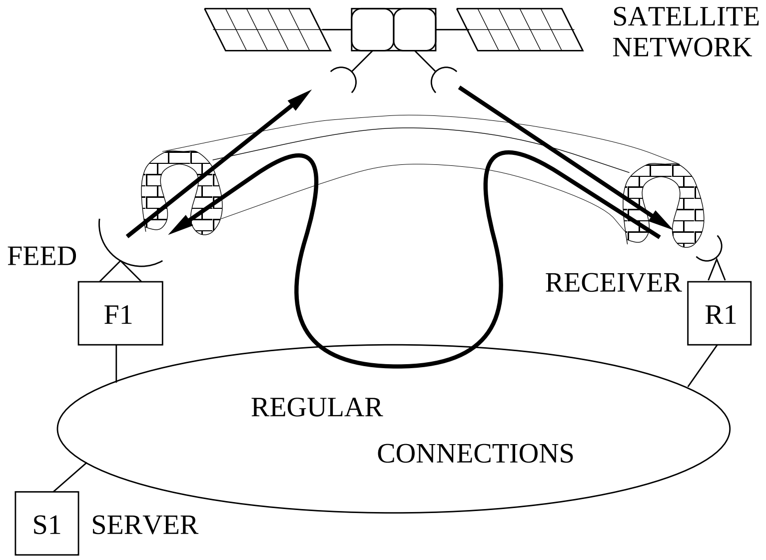


Figure 2.1: Tunneling

2.1 Existing Proposals

The first approach, tunneling, sets a tunnel to emulate a bidirectional link over the satellite. It hides the UDL without modifying the routing protocols. The second, RPM, directly modifies the routing protocols to take into account the unidirectional nature of the link. It explicitly specifies a link as unidirectional.

Basically, both approaches aim to keep the changes needed to a minimum. They are therefore trying to give a quick short term solution allowing the Internet to use this particular type of link and may not be optimum. The considered protocols are RIP, OSPF and DVMRP.

After presenting the different approaches and their proposals we discuss around the advantages and disadvantages of tunneling versus RPM. Long-term and short-term relevance is considered, and we conclude whether tunneling or RPM should be used to consider the problem of unidirectional link routing.

2.1.1 Tunneling

Tunneling is a mean to construct virtual networks by encapsulating the data. Broadly speaking, data is sent by the network layer instead of the data-link layer. 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 actual one. This can be used to support UDLR without any modification of the routing protocols. Implementors should be aware of some pitfalls as described in [28] and discussed in [8]. For more information about tunnels, we recommend to read [31], [14], [22] and [6] which describe possible techniques to implement tunnels. [22] describes a tunneling technique used to skip routers which cannot handle the required behavior, while [14] presents another technique rather used to carry different protocol families.

The tunneling approach adds a layer between the network interface and the routing software on both feed and receiver sides (or between some intermediate gateways), resulting in the emulation of a satellite BDL where only an UDL 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 satellite BDL. See figure 2.1.

Basically, routing traffic is sent from the receivers to the feed on the virtual link and is later captured by the added layer interface which encapsulates it and sends it on the actual reverse link. The feed station then decapsulates it and transmits it to the routing protocols as if this was coming from the virtual satellite link. At the sender, the terrestrial BDL is never used to send traffic.

There are several groups proposing to use tunneling to support an UDL on top of a BDL network. Current proposals are from Wide [1], The Aerospace Corporation [26] and Hugues Research Laboratories (HRL) [32]. The terms used are those introduced in the various proposals and will therefore not be defined in this report. The presentation of the proposals make them look different one from another at first sight, but their careful study shows that they are indeed very similar. One of the differences between the proposals is whether data traffic is sent, indeed encapsulated, or not on the virtual reverse link.

WIDE This proposal has been conducted in Japan at Keio University in collaboration with the WIDE Project (WISH working group) which aims to enhance domestic communications using satellites in Japan. In [17], WIDE describes a "Virtual Interface" (VIF) as described in [8] between the data-link layer and the network layer that can separately use unidirectional and bidirectional links. This strategy is based on the setting of a tunnel from the receiver to the feed which establish a virtual reverse satellite link and elude the routing protocols. Two bidirectional paths connecting a feed to a receiver are resulting from this approach: the actual terrestrial BDL, and the virtual satellite BDL. This first relies on the mutual exchange of the IP addresses of the two possible interfaces at each sides, i.e. the satellite UDL and the terrestrial BDL. Basically, this exchange is performed by a static configuration but it could also be done dynamically using their Dynamic Tunneling Path Configuration. The feed periodically sends its BDL and UDL IP addresses on the UDL. A new receiver then sends its own IP addresses to the feed. This automatically establishes a tunnel and canceled it once the feed is down. The system is much like connection oriented. There might be scalability problem here because this connection oriented mechanism requires to keep state at the feed on a per receiver basis. Routing protocols basically see two BDLs links, but routing tables are configured in order to route traffic originated at the receivers via the terrestrial BDL. Therefore, only the routing traffic needed by routing protocols is encapsulated. The aim of tunneling is therefore only to make the routing protocols at both side of the satellite link to be aware of it. Encapsulating the routing traffic would obviously still add overhead, but the authors claim that this is negligible since the amount of routing traffic is not very important compare to the amount of data traffic. This preference of the terrestrial BDL over what the routing protocol sees as a satellite BDL is given by setting it to a higher cost than the terrestrial link. At the feed, the satellite UDL must be given priority over the terrestrial BDL by setting it to a lower cost than the terrestrial BDL. The traffic will always pass through the cheapest link. Those links then appear asymmetric to the routing protocols. What value should be given to the metric of the preferred link remains an open question. This obviously depends on the routing protocol under consideration and on the point of view of people. The system has been implemented and is currently being evaluated on a domestic test-bed using a real satellite UDL, on which RIP, OSPF and DVMRP are working. The UDL behaves like an Ethernet. Another Internet-Draft about dynamic tunneling should be published soon.

Aerospace Aerospace's "Virtual Internet Packet Relay"(VIPRe) differs from Wide's tunnel. The routing protocols see the satellite UDL as if it was BDL as in Wide's, but it only sees one bidirectional path. The virtual reverse satellite path is an one-way tunnel or "relay" between two gateways called "Split Virtual Interfaces" (SVIFs) which respectively aim to encapsulate or decapsulate packets. This tunnel is set up on top of the existing terrestrial BDL which is never used directly. On the contrary of Wide's proposal, no distinction is made between routing traffic and normal traffic since there is only one logical path. All traffic is captured and then passes through the tunnel. One could say this lead to overhead but the authors argue that this was the best way to make the things work, this maximizing software compatibility and minimizing changes needed at the UDL endpoints. Unlike Wide's proposal, this doesn't require modifications of routing table entries in order to give one path the preference over the other. The advantage is that upper protocols which assume that links are BDLs can operate unmodified, but this architecture fails to handle the asymmetric nature of the virtual BDL. The cost of the reverse link must then be carefully set to reflect the high cost of the satellite link in term of delay and the bandwidth of the underlying structure of the terrestrial path. The authors introduced a new type of ICMP message called "Relay advertisement" which aim to exchange IP addresses before the tunnel could be set. It advertises the up-link SVIFs address to the potential receivers. This provide for a dynamic discovery of feeds.

Hugues Research Laboratories (HRL) Originally, HRL has designed a product called "DirecPc" (a description is given in [21]) which provides Internet connectivity for end-users only via a satellite UDL and a dialup back channel. Other hosts in a subnet couldn't be joined because the feed is not aware of their identities. A tunnel is set between the receiver called "Hybrid Host", which is a multi-homed host with two IP addresses, and the "Hybrid Gateway" near the feed. Then, they have designed a Direct Broadcast Satellite (DBS) system [32] using their DirectPc system that provides instant Internet connectivity to mobile computers, but this system doesn't allow dynamic routing. HRL's proposal for UDLR is apparently enhancing this system allowing dynamic routing by creating a virtual network which emulates a bidirectional network. It has been implemented in a small test-bed, using RIP and without turning on dynamic routing. It seems that OSPF has never been tried. Since there are no technical details about their approach, we won't go deeper in the description. Anyway, it looks much the same as Wide's. The reader may take a look at [30] to have more information about HRL's proposal. As emphasized in [21] and due to encapsulation, packets going from the Hybrid Host are always routed via the hybrid gateway. This could lead to a longest path.

Subnetwork connectivity using DirecPC The system described in [21] and designed at Berkeley provides DirecPc extensions in order to support a subnet with multiple hosts connected to it since this was not possible with the original system. There is still a multi-homed Hybrid host, this time connected to a "Hybrid Subnet". The system is similar to that employed in IP-Mobile. There is a Home Agent somewhere between the sender and the feed that captures traffic and ensures its correct delivery by tunneling it to the Hybrid Host over the satellite. Traffic is therefore encapsulated in both directions, but there is no mention of routing traffic. There is therefore no dynamic routing. Those are therefore not relevant solutions to UDLR.

2.1.2 Modification of Routing Protocols

RPM is the alternative to tunneling and is proposed by the INRIA. The INRIA's solution as it is described in [10] is based on modifying existing routing protocols to remove the BDL assumption. Protocol are therefore explicitly aware of the actual underlying topology of the network, as opposed to tunneling. Receivers detect dynamically the presence of feeds and send routing messages back to them via regular connections.

The modifications needed are not the same depending on the nature of the routing protocol under consideration, basically LS protocols as OSPF or DV as RIP. In the first case, all existing routers in the network have to be updated by the new version of the protocol. Gracefully updating all the routers without disrupting the network is feasible if we allow the two versions to cohabit together for a while, but the cost is necessarily higher than in the second case where only those routers locating at the edges of the UDL require an updating.

Three Internet drafts describe what changes INRIA advocates for some of the most popular routing protocols, OSPF [11], RIP [12] and DVMRP [9].

2.2 A comparison between tunneling and RPM

2.2.1 Static vs dynamic routing

Each feed basically needs to know which are the subnetworks it can access via the satellite link. Static configuration is fine and could be made carefully as long as the topology is not too complex, but it cannot handle large number of receivers and shouldn't be considered any longer since the actual topology is likely to be more and more complex. Dynamic routing must be supported to manage changes in the topology and allow a large number of receivers to benefit from this facility. DirecPC and its Berkeley extensions are therefore not relevant to UDLR, but both tunneling and RMP approaches in theory allow dynamic routing even if dynamic discovery of feeds and subnetworks isn't yet implemented in all proposals.

2.2.2 Routing Efficiency

Overhead There isn't a lot of overhead due to tunneling since only routing traffic is using the tunnel. This doesn't represent a significant amount of traffic, and encapsulation itself doesn't really lessen performances. Aerospace allows data traffic through the tunnel but only because it makes the things consistent. Efficiency is therefore preserved in tunneling except for DirecPC and Berkeley's extension to DirecPC where encapsulation is imposed. We can conclude that both approaches could route data through the terrestrial path without encapsulation.

Sub-optimal Routes Tunnels are hiding the actual topology of the underlying network and are therefore leading to sub-optimal use of routes. Tunnels are static and don't reflect changes, neither do they reflect the actual distance between the two edges. Routing protocols cannot find an optimal route if false distances are voluntary announced. Moreover, future enhancements in the Internet such like quality of service and reservation for which tunnels are unlikely to well interact with tunneling. Tunnels basically are a kind of stupid abstraction on top of a clever network of routers. RPM on the contrary dynamically exhibit the actual topology of the network so that the most accurate decisions are taken. This advocate the use of RPM rather than tunneling.

2.2.3 Easy configuration

In the RPM approach, the routing protocols are aware of the unidirectional nature of the satellite link but this is hided in the tunneling approach where the routing protocols see two BDLs. In order to avoid to encapsulate traffic on the reverse virtual link, tables are configured by setting metrics. The virtual bidirectional link therefore appears to be asymmetric. But this is not a real asymmetry; it is only a trick since one BDL and has to be systematically preferred over the other. The RPM approach explicitly handles UDLs and could therefore give this choice to the routing protocol, likely according to applications requirements. This approach could therefore allow to enhance routing since many links actually have an asymmetric bandwidth. But this should rather be seen as a long term advantage. At the receiver, the choice between the virtual satellite link and the terrestrial link is made automatically and statically in tunneling. In tunneling, setting the right metric is an important question without consensus. It is trivia for DV protocols, but finding the right value in LS protocols is less obvious. In RIP, we are limited in the scope of possible values. How to select the cost in very large network? The side effect is that we must set asymmetric metrics to avoid that the virtual back-channel is used for sending something else that routing traffic. There is no need for metric setting in the RPM approach since the routing protocols know there is no satellite back-channel. Both approaches are therefore auto-configurable.

2.2.4 Scalability

A feed will likely be connected to a large number of receivers geographically distributed, this leading to implosion of state on the feed side. The problem is therefore the same for both approaches. One solution is to increase the number of feeds so that each feed deals with a particular set of receivers. We should note that receivers connected to numerous satellites at the same time seems out of point at the present time due to present technical limitations. This configuration would rather require to have several dishes on the receiver side, indeed one for each satellite. Concerning tunneling, we could set up point to multi-point tunnels from one receiver to multiple feeds as described in [19], but the outer endpoints have to be specified, therefore leading to a limited number of possible endpoints at the feed. The problem of scalability still remains. There is nevertheless no mention of point to multi-point tunnels in the current proposals addressing UDLR, except the one from HRL.

2.2.5 Transparency

Dangers in tunneling Tunneling is dangerous because it is a hack. There is an obvious lack of transparency which could cause problems to implementors if they are not aware of what is exactly going on. This is all right as far as experimentation and temporary solutions are concerned since tunnels have to be carefully manipulated. Tunneling indeed provide an excellent test-bed for new protocols experimentation and for a smooth transition from one protocol to another, but this should be seen the other way round for a permanent solution. [28] describes "mutual encapsulation" which is a potential pitfall. Others pitfalls such like fragmentation and black holes, and ways to avoid them are described in [8] and the RFC describing tunneling [31] but the lack of standardization for a particular tunneling technique doesn't help in this. Moreover, there is a risk that UDLs are set up in an anarchic fashion. There is therefore a need to agree on a conventional model to deal with tunneling. Attempts of standardization are given in [14], [22] and [6] that describe a general purpose mechanism for encapsulation, but using a particular approach of encapsulation has not yet been required. Moreover, [14] doesn't address mutual encapsulation. Those points recommend to use tunnels with care, to agree on an existing standard and rather favor their use for the very short term.

Dangers in RPM Protocols modifications might result in having a plethora of ad-hoc versions that could all differ in a way from the original version, each one addressing a particular problem. Such new versions could abound in the Internet community. Moreover, some upper level protocols and softwares closely interact with and rely on the underlying topology. Those might only perform well on top of BDLs. Allowing UDL would involve to reconsider them. Even if there is a need for doing so, this cannot be done in the short term without pitfalls. There will nevertheless be some reasons for which some modifications would be needed if we consider other areas of research (eg mobile IP). It seems then relevant to update existing routing protocols in a whole and to propose a new standard where modifications allowing unidirectional routing should be merged to those already required. Current research in modifying protocols are then likely to find an useful issue and should be enhanced. Inria's proposal is therefore a good practice in order to design next generation routing protocols. This rather balances RPM for the long term.

2.2.6 Inter-domain routing

Receivers are likely to be in different ASs (see section 2.3.1) since the footprint of the satellite is very large. This has to be dealt with BGP which is the current protocol dealing with inter-domain routing.

Inter-domain routing involves routing across several ASs. All of them are unlikely using the same IGP inside their AS since there exist several standards. This means that different metrics, and even different kind of protocols are used. Path selections therefore obey various criteria according to the local AS protocol and preferences. This obviously bring a bit of confusion for inter-AS path selection.

We will consider this question in much more details since it has not yet been considered by both approaches, although it is likely to be an important issue. We will present our propositions in section 2.3.1, but we can already say here that the question of inter-domain routing cannot be treated as easily, although some people pretend that inter-domain routing is not a problem to the use of UDLs.

2.2.7 What solution should we choose then ?

The discussion shows that the problems which need to be resolved are indeed much the same in both approaches although they are very different at first sight. Basically, further consideration need to address the question of scalability when feeds and receivers are going numerous, multi-casting and inter-domain routing.

RPM not to mention tunneling, seems only viable as long as UDLs remains the exception rather than the rule. Applications requiring such connectivity are not common at the moment, but satellite operators are likely foreseeing a new kind of application using this facility. Designing new routing protocols might be necessary if this is becoming of widespread use.

It therefore looks like the RPM approach is the one to investigate in order to produce a long-term valid solution to unidirectional routing. Tunneling seems to be only relevant to the short-term, but is rather a "do-it-yourself" or ad-hoc solution not well suited to address the very soon foreseeing increasing demand on unidirectional links. Tunneling might soon or later be banished and therefore serious investigation in RPM is recommended in order to make an experience in this area. RPM can also temporarily provides as tunneling a solution to the current problem of unidirectional routing but its start-up could take a longer time.

We therefore focus on tunneling for a satellite UDL because such solutions are operational quickly but we recommend the modification of BGP for the issue of inter-domain routing. For the All-UDLs case, we favor modification of routing protocols since tunneling is not really adequate for the long term, but we will tackle this question in subsequent chapters.

The reader should note that proposals which appeared in this paper have been designed with satellite in mind, but those could possibly be used in others architectures.

2.3 The case of Inter-AS routing

We first outline the differences between an IGP and an EGP for the purpose of the discussion before presenting slightly BGP-4, the current EGP in the Internet. After that, we will give our view of the problem of locating BGP routers when using a satellite UDL to reach receivers located in various ASs. There are basically three architectures. Feeds and Receivers might form a special AS called "satellite-AS", the receivers are external gateways of the feed's AS, and finally, feeds and receivers remain in their own AS.

2.3.1 What are AS, IGP and EGP ?

There exist two kinds of routing protocols: internal gateways routing protocols (IGP) and external gateways routing protocols (EGP). OSPF [20, 16] and RIP [15, 18, 16] are examples of IGPs while BGP [24, 23, 16] is one of an EGP. IGPs allow routing within an Autonomous System (AS) while EGP provide inter-autonomous system reachability. An AS is a subset of the overall Internet where common policies and the same routing protocol are running. The purpose of the partition of the Internet into ASs is to mitigate the growth of the Internet. This limits the amount of information kept in routing tables.

We should notice that all the protocols considered until now by both approaches are intra-domain routing protocols (interior gateway protocols - IGP). This requires that sender and receivers are all located within the same AS. Well, satellites have a large range of coverage. Sender and receivers might well be located on distinct ASs, and different receivers might as well be located on widespread or distant ASs. Considering that all receivers are in the same AS as the sender is possible and is covered in the above sections, but this is a limiting factor to the use of UDLs over satellites. We therefore need to consider a solution that makes this particular use of UDL possible over a set of ASs. In others words, we need to consider an External Gateway Protocol (EGP) running on receivers in order to access subnetworks connected to the receivers. BGP is the one in current use; we are therefore focusing on it.

There is an IGP running within each AS. Its aim is to compute internal routes inside the AS whereas the EGP is running on each external gateways and compute routes across the entire Internet. The EGP provides each external gateways of the IGPs with reachability information about all accessible networks.

2.3.2 A quick presentation of BGP

We only present the characteristics of BGP-4 relevant to this section. For more information about BGP-4, the reader should take a look in the specification [24, 23] or in [16] for a general discussion.

BGP-4 is an inter-domain routing protocol. Unlike current routing protocols, BGP runs over a reliable transport protocol, TCP. This greatly simplify its implementation because procedures usually executed by transport protocols aren't incorporated in the routing protocol.

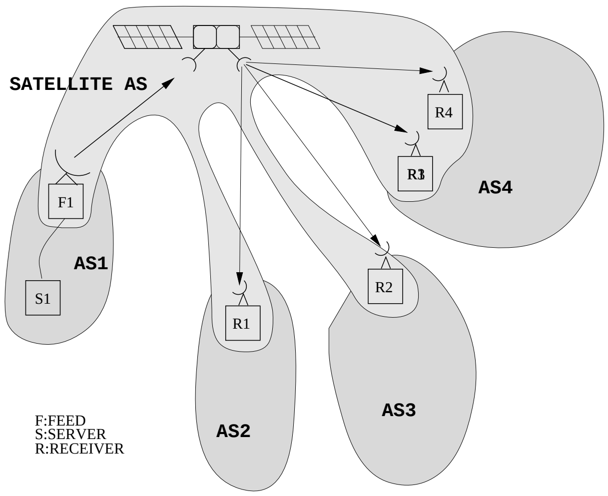


Figure 2.2: Transit Satellite AS

Basically, there is a TCP connection between two BGP speakers willing to exchange reachability information. This also means that BGP assumes a graceful close. Liveliness of connections is ensured by Keep-Alive messages sent periodically. Once the connection procedure has fully succeeded reachability information can be exchanged.

The use of TCP will probably cause a scalability problem when using a satellite UDL because it requires to keep state at the feed on a per receiver basis. Routes are implicitly becoming invalid when a TCP connection is closed. Moreover, upon error detection, the TCP connections involved are automatically closed.

If should be noted that the use of TCP over satellite is another subject over investigation, out of the scope of this present study. Although, we could say that having BGP using TCP might well bring some more problems. Actually, the slow-start algorithm of TCP already doesn't perform well over a satellite BDL due to its long delay bandwidth nature. The round trip delay should be smaller.

BGP-4 is using the technique of Path Vectors which provide the description of valid paths, i.e. traversed ASs in an EGP. Nodes therefore maintain a list of ASs traversed for each valid route. Those are obtained using flooding on start up, and update are sent whenever required. Only routes actually used by a node are advertised to neighbors.

A BGP speaker first need to be aware of the networks it can reach via the satellite link. The problem is much the same than for OSPF or RIP. Networks discovery is done on start up. Once nodes are discovered, pairs of BGP speakers at both side of the satellite link establish a TCP connection. This TCP connection should be established using the satellite link in one direction and using a terrestrial reverse link for the return path. The current specification of BGP requires that traffic sent from one AS to a neighboring AS takes the same route from that taken by the traffic originating in the neighboring AS. This has limitation in face of UDLs since the reverse path is likely different, but the specification leaves full discretion to administratively assign a degree of preference to prefer a path leading to a particular destination. This degree could range between 0 and 2^{31} which is much more than the metric used by RIP. Because that can be done easily, people argue that inter-domain routing is not a problem to the question of an alternative UDL on top of a BDL network. But the question of TCP and the location of BGP routers remains an open issue.

If BGP is running, that implies to keep large tables and requires much more CPU. We are therefore loosing part of the benefit of low cost investment by receivers as depicted in the previous section. This should be balanced against the design of a new routing protocol supporting UDLs. Since designing a new protocol is rather seen as a long term solution to the question of UDLs, this is discussed in the following chapters.

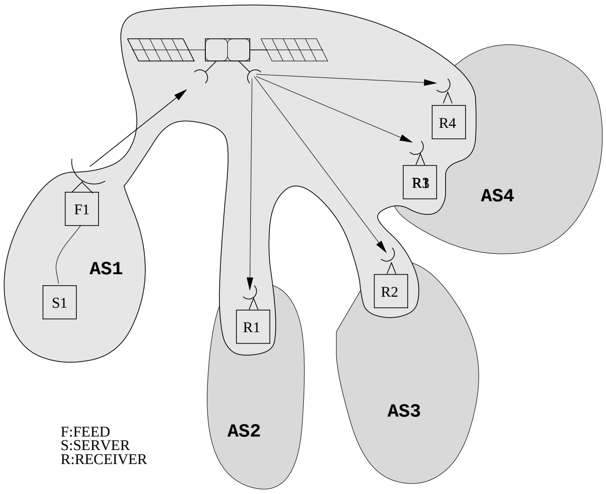


Figure 2.3: Receivers are External Gateways of Feed's AS

2.3.3 Feed and Receivers belong to a "Satellite AS"

In that configuration, the feed and the receivers belong to the same "satellite-AS" as shown on figure 2.2. We therefore need to run an EGP at each external gateway of the AS, indeed all nodes of the satellite-AS. That means that every feed and receivers are external gateways, possibly connected to distinct ASs. This AS is nothing less than a transit AS, and could be called "transit satellite AS". The IGP running in our transit satellite-AS doesn't need to compute any route since all nodes within this AS are external gateways and are directly connected to the feed. It nevertheless needs to provide each external gateway with the information about reachable subnetworks.

All routers located in the satellite-AS are using BGP. This is called "pervasive BGP" in the specification [23]. There is no need to have any interaction between BGP and the IGP since all routers already have knowledge of all existing BGP routes. No BGP routes are imported into the IGP. The task of the IGP is only to transmit reachability information from the external gateways located on the satellite-receivers back to the feeds.

All BGP speakers located in the same AS need to give a consistent view of the AS to the outside. For that purpose, it's recommended that all those BGP speakers have a BGP connection directly one to another in order to keep consistent information among them. Basically, a consistent view of the interior routes is generally given by the IGP running in the AS.

All links are UDLs coming from feed to receivers, thus leaving us with the design of a special purpose IGP that would ensure consistency. This mean that we need a way to provide this reachability information from the receivers to the external gateway running at the feed. Providing this information back to all gateways located at receivers is trivial once the feed gets it. The IGP could look like the protocol Hello, where feeds inform all stations connected to the satellite that they are ready to send traffic.

The problem of this architecture is that the graph representing this AS is not connex. This involves the need to transmit routing traffic back to the feed through others ASs. One easy way to do it would be to establish a TCP connection through other ASs although this might not scale well if receivers are too numerous. This solution wouldn't involve any tunnel setting neither protocol modification, and doesn't make any assumption on the IGP running within the transit AS.

Links would appear to be bidirectional to each BGP speaker. Basically, the design of BGP allows to administratively set preferences to avoid traffic passing through some links. BGP wouldn't therefore advertise routes from the receivers to the feed which are not optimum. BGP doesn't require any modification once it can get the reachability information.

It should be noted that we don't need to run a protocol such like BGP in order to provide reachability between ASs if the ASs connected to the satellite receivers don't allow transit traffic through those ASs

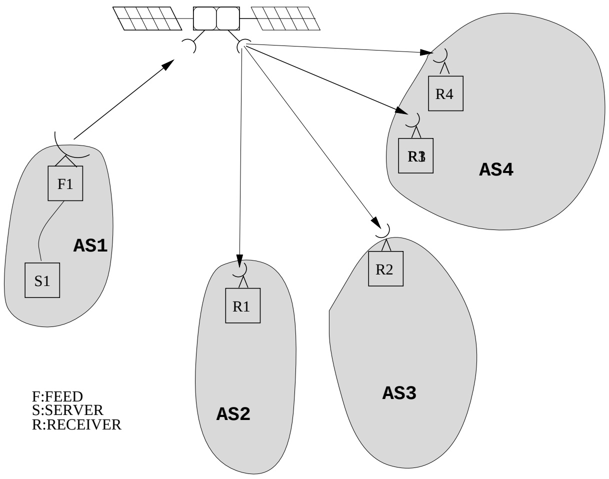


Figure 2.4: Receivers and Feeds are External Gateways of their own AS

(i.e. stub AS, or multi-homed AS refusing transit traffic). Other protocols such as EGP [1] would be efficient enough, but we nevertheless need to consider BGP as the most general case if only one AS could accept transit traffic. Moreover, since we need a return path to the sender, we should likely consider those ASs as transit ASs anyway.

2.3.4 Receivers are external gateways of the feed's AS

All satellite receivers might be located on a distinct AS than the feed. We are nevertheless considering receivers as part of the same AS than the feed, as shown in figure 2.3. Any station willing to receive data from the satellite has to register itself as a new member of the feed's AS deserving external networks. Receivers are therefore external gateways located on two ASs.

This configuration implies that the feed is not located on an external gateway. Receivers and feeds are under the management of the same IGP, likely OSPF or RIP. This configuration is close to the one presented in the working group, with respect that routing traffic of the IGP requires to transit through some others ASs in order to integrate satellite receivers in the IGP. IGP doesn't allow that, then the RPM approach might not provide an easy solution. On the other hand, tunneling should easily do the job.

This configuration could remain transparent to BGP, but it would be better to set preferences to prevent BGP using the reverse connectivity of the satellite receivers.

2.3.5 Feeds and Receivers are external gateways of their own ASs

In this architecture, all feeds and receivers remain in their respective AS as it could be seen in figure 2.4. Any traffic through the satellite would therefore cross a border between two ASs. This configuration requires that all feeds and receivers are running the current EGP, then BGP.

This involves a modification of BGP in order to support UDLs going from one AS to an other. This could be done by setting a tunnel or modifying BGP to take directly UDLs into account. This might be trivial since preferences to some paths over the others could administratively be set. The same debate between protocol modification and tunneling may apply here.

[1] EGP is as well an inter-domain routing protocol, the ancestor of BGP

Chapter 3

Long term applications of UDLs

In this chapter, we study the question of UDLR for the long term. As a general matter, the right answer depends on the range of applicability of UDLs. We are foreseeing their use in networks of limited size and scattered across the Internet at the same time. In others words, they are likely to be of general use.

After studying in more detail the three different architectures of UDLs we have presented in chapter 1 (page 7) we notice that a single general solution could apply to all three when considering a long-term solution. The same questions are indeed emerging, and we should consider those before designing our proposal to the long term.

As a reminder, in the first architecture, there is only one UDL, likely a satellite UDL from a feed to its receivers. In the second, there are BDL islands interconnected by UDLs. In the third case, all links are UDLs within the network under consideration.

In this chapter we aim to answer to some questions like “Would current protocols solve the problem, do we need to modify them, or do we need to design new protocols?”, “Do we need one general protocol addressing the question in its whole or several special purpose protocols?”, and “ What type of protocols would it be? Do we need a full or partial knowledge of the topology?”

In section 3.1, we present the overall picture of *a general long term solution* to all three cases. For this purpose, we first show why our solution should be an IGP. The best solution is effectively to *design a new protocol* as we describe it in section 3.2 and not to modify current protocols. Section 3.3 considers *the characteristics of UDLs* to take into consideration for the design of a new protocol because it might affect the behavior of routing. We consequently study in section 3.4 known routing techniques currently in use in the Internet, i.e. Link State, Distance Vectors and Path Vectors which is an enhancement of DV. None of them is applicable as is. From this point, we discuss in section 3.5 what are *the expectations of a new protocol* in an UDL network and briefly propose a new technique for routing in UDL networks. This technique is described in the following chapter.

3.1 A General Long Term Solution

According to the definition of an IGP and an EGP, as written in section 2.3.1, and the range of a satellite link, we will unlikely design a general purpose protocol that take into account UDLs for both IGP and EGP at the same time.

The one-UDL-only case was discussed in the last chapter. This is solved by the UDLR working group and by BGP once receivers are located in different ASs. This architecture won’t generalize in the Internet, otherwise it transposes in the second and third case of UDL. If UDLs are not going to spread, that means that the debate is closed and that the agreed solution is tunneling.

The BDL-islands architecture is indeed very similar to the all-UDLs architecture if we consider islands as single nodes of an all-UDLs network. Of course, BDLs islands could be seen as ASs interconnected by UDLs. This case is therefore blending with the design of an inter-domain routing protocol (EGP) in the family of BGP that implicitly takes into account UDLs. On the other hand, the All-UDL architecture could be seen as an AS where all links are UDLs, with the assumption that the graph representing the network is connex, that is, all nodes can reach each other without relying on inter-domain routing. Moreover, if there is a certain amount of true BDLs in an all-UDLs network, we might group them in BDLs islands and therefore only consider their interconnection via UDLs.

Basically, and whether we design an EGP or a IGP, the unidirectionality of links will have to be taken in a very similar way. Our feeling is that both an unidirectional EGP and an unidirectional IGP will find their foundations in a common solution. We are therefore focusing on a general long term solution.

3.2 Why do we need a new protocol?

Tunneling, as described in the previous chapter, aims to give a quick and evanescent answer to a short-term problem (i.e. one satellite UDL only), with a minimum of changes. This solution is obviously not optimum, but is good enough in face of sparse UDLs and should be balanced against the cost of designing new protocols and spread their installation across the Internet, which won't be easy and cost more than the potential benefit. The question should be seen the opposite way round if we consider UDLs as being of common use in the middle or long term future, as it is foreseen.

Solutions based on tunneling will not work because tunneling assume the existence of a bidirectional back-channel in order to establish the tunnel. This assumption is not longer true in All-UDL network. This could directly lead to deadlock if the back-channel is itself unidirectional and therefore requires the setting of a tunnel.

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. Of course, we are trying to use the experience built from those existing protocols, particularly techniques relevant to UDLs.

Another argument that is weighting for the design of a new protocol is that many routing protocols anyway require modifications or extensions to remove asymmetric link assumptions. We have already seen that considering a BDL as being two UDLs allow to solve this problem at the same time as unidirectional routing.

3.3 Characteristics depicted by UDLs

The obvious characteristic is that communication is not possible directly in both ways. This gives rise to various considerations, like how acknowledgments are processed, how nodes are aware of their neighborhood and of the network topology. Because of those characteristics, some un-practicable solutions with BDL networks might prove to be of great interest in UDL networks.

Acknowledgments Acknowledgments of routing traffic are generally sent hop-by-hop. Because links are unidirectional, acknowledgments of routing traffic have to follow a different path to that of the outward traffic instead of simply being sent back on the same link. That would add overhead compare to BDLs. It would therefore be useful to avoid sending ACKs for routing purpose whenever possible.

Asymmetry of links Each node aims to find the cheapest route to reach destinations in the network. Routing protocols generally assume that links metrics, therefore paths metrics are symmetric in order to compute those shortest distances, based on the fact that node A can get to not B by the same path that B uses to get to A. Because this is not longer true, as figure 3.1 shows, routing protocols require important changes and even more messages might be necessary to obtain the same knowledge. Moreover, obtaining the full knowledge of destinations is itself harder because a node cannot even assume that it can get to a node that can get to it.

Neighborhood reachability Acknowledgments allow pairs of routers to be aware of their connectivity or of the failure of neighbors. This is a necessary requirement which ensures that the neighbor can safely be considered operational and capable of providing reliable reachability information. UDLs require an other mechanism to ensure peer to peer reachability. The successor of the link can easily be aware of the failure of links coming from its predecessors if it doesn't get periodic traffic from those. But there is no straight mechanism to tell the predecessor about such failure. It would still keep sending routing data that would necessarily be lost. Moreover, a node is unreachable in a directed graph when all its incoming links have failed. This means the graph is not connex anymore, as well if a reachable node has no successors. There is no guarantee for this node that data it sends is correctly received. Therefore, a node might be considered "out of order" even if it still have valid outgoing links and it should stop sending data, but to say it cannot be reached. We conclude that the successors consider the predecessor of a failing link as unreachable and that they have the responsibility to inform it of the failure.

Circuit Detection Determining a path should be seen as determining a path to get to the destination and one to go back to the sender. If one of those two paths doesn't exist, that should lead to the conclusion that the peer is unreachable. Links being UDLs, the reverse path might not be the same than the outward path. Textually, a circuit is an itinerary that brings back to the departure point. That is exactly what we need. The return path might sometimes take a common part of the outward path. For instance, figure 3.1 shows a directed graph. A circuit originated at A aiming to reach Y will have to pass two times on link BC. If the circuit was passing through nodes $ABCEYBCDA$ and the destination was not on the circuit $BCEYB$, we would speak about a loop. Detecting circuits could therefore inform the local node of the reachable nodes and ensures their reachability too. Likewise, if there are no circuits, the graph is not connex and the node shouldn't transmit data even if it knows it is reachable. There exist algorithms in the literature that have been designed in order to detect and avoid routing loops in the Internet. Those routing loop detection and avoidance algorithms could be used in UDL networks and transposed to circuit finding.

3.4 Reviewing of known techniques

Since existing protocols might well provide foundations for a UDLR protocol, we need to point out the applicability and limitations in face of UDLs of the current routing techniques in use in the Internet. Those techniques essentially are *Distance Vectors* (DV) and *Link States* (LS) which are used in IGPs, and *Path Vectors* (PV) which is currently used in an EGP. According to section 3.1, the study of an EGP technique is relevant. We highly recommend the reading of [16] for those readers not familiar with the current techniques.

This study shows which of link states, distance vectors, path vectors protocol, or something else best suits all-UDLs network. It is outlining their advantages and drawbacks, therefore what modifications are required. Basically, this demonstrates that none of them can be implemented as is.

3.4.1 Distance Vectors

The DV technique is very simple, easy to implement and understand. The computation of routes is distributed, that is, each node is directly using the routing data to compute shortest routes and only forward those. The philosophy of DV is to maintain at each node a table containing the next hop on the shortest path from the local node to each destination in the network. In the basic version of DV, the only information recorded in the table is the destination, the cost of the best path and the next hop. This means that each node only has a local view of the network and has no knowledge of paths that do not pass through it. Since there is only one entry per destination, this technique doesn't provide for multi-path routing.

An upper bound called infinity is fixed for distances. The value of infinity has to be rather small in order to avoid routing loops in case of inconsistency, since inconsistency might lead to a perpetual incrementation of the distance through the network. Unreachable destinations have there distance set

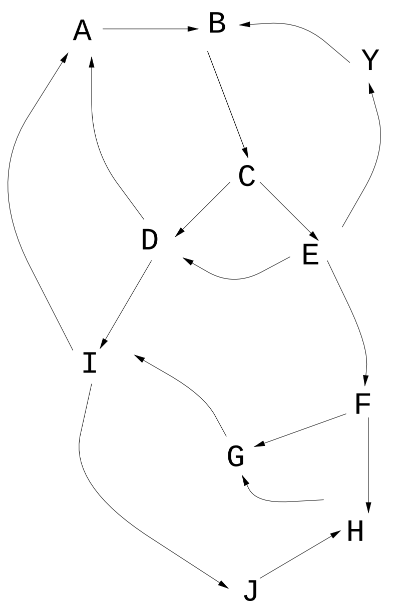


Figure 3.1: An instance of UDL network

to infinity so that distances can never exceed this fixed cost. This process is called counting to infinity. This obviously limits the size of the network and the accuracy of the metric.

In addition, only one metric is used in standard DV protocols. Some enhancements might of course allow to use more than one metric, but the system would have to maintain a table for each metric. Several metrics provides a more optimal path selection, according to the application requirements. Moreover, DV protocols are seen as very easy to implement and flexible if limited to its basic facilities. But simplicity is at the cost of efficiency. Many improvements can bring a higher efficiency, but this leads to a tradeoff between ease of implementation and efficiency.

Basically, the algorithm proceeds likes this to compute those tables in a BDL network: each node periodically advertises to its neighbors the shortest distances it has to reach each destination in the network. At the beginning, each node only transmits its local route, i.e. *from me to me, cost=0*. Upon reception of DV advertised from a neighbor D, a node I adds to the cost of each destination it received its cost Cid to join D and considers D as the next hop on the way to those destinations. Destinations that doesn't exist in the local table are added in the table, and destinations already in the table with higher costs that the new ones are amended with the new cost. DV containing changes are thereafter transmitted to all neighbors that proceed like this as well.

DV could basically be summarized by saying that if D informs the local node I that it can join a third party A, then I can come to the conclusion that it can reach A as well via D. This is not true if links are UDLs, first because the cost is not the same in the two directions and because D is not the next hop on the way, unless there exists a direct link in both directions and is probably not on the way at all to reach A. The information *D can reach A* received from D at node I therefore doesn't help I to deduce what is the cost to get to A.

We have seen in section 3.3 that there necessarily exists a circuit connecting two nodes. We can therefore deduce what are the destinations *I can reach* from *the nodes that can reach I*. But DV would be applicable to UDLs only if it is possible to deduce the cost of the path.

Changes Required to DVs The concept of DV implemented as it is in UDLs networks could easily provide the reverse information from what is required, i.e. the cost to get from node A to I instead of the cost from I to A. We cannot do anything with this information but transmitting it back to node A in order to be useful. Another problem, once the information is back to the initiator of the DV, i.e A, is that we are losing the name of the next hop B on the path unless we have explicitly recorded it.

Adding of a new link We first make the assumption that all existing nodes in the network have their tables filled with the cheapest distances to each destination. Obtaining at I from the predecessor D its distances to all destinations could be useful if I already knows the cheapest cost Cid to reach D. In fact, I can conclude that if D can reach A with a cost Cda, then I can reach A with cost $Cid + Cda$. The question which remains is how I get the distance to get to D since $Cdi \neq Cid$. However, it is right to add Cid to Cda, but this doesn't necessarily produces the cheapest cost to get to A even if Cda is the cheapest from D to A. Figure 3.1 clearly shows that. Then, we still haven't found DVs.

Forwarding of DVs Basically, we need to use DV in a different manner. We can easily get the distances from the others to I if all nodes forward DV by adding the cost of the link where it received the DV. In order to be useful, the information A can reach I has to be forwarded to A. This has to be done using flooding since it is not possible to know successors when starting from scratch. Otherwise, we can send it via the cheapest path. Nodes on the way might used the information to establish later their cost to reach nodes. All transit data should be recorded in the table for doing so.

Starting from Scratch This consists for each node to discover who are its successors. DVs are forwarded until the initiator of the DV whenever shorter distance to get from a source to the local node are found. When the process is over, each node knows who it can reach with which cost. We assume that nodes are initiating the topology discovery by sending their local cost, as described in DV. I would send *from I to I cost=Cii=0*. From this information, A adds the cost Cia and forwards *from I to A cost=Cia next-hop=A* on each outgoing links. B would then forward *from I to B cost=Cib next-hop=A* and *from A to B cost=Cab next-hop=B* and so on. Since flooding distances from each node to everywhere is necessary, we see that we flood more information than the Link State technique does. Providing in an UDL network the same information than in a BDL network is therefore more costly, in term of number of message, states in the node, and possibly CPU since nodes do not only add their cost to the DV as it was done in BDLs networks.than in a BDL network, and even more than with the LS technique for a worst result.

The nature of DV is in fact highly modified and doesn't look like DV anymore since its main characteristics are changed. At last, and likely not least, the mechanism is very complex to find new routes upon failures. We therefore do not recommend the use of DV in UDLs networks. We tried to find out solutions starting from the concept of DV without any success due to the high degree of complexity of this mechanism in an UDL environment. Further work might of course demonstrate the applicability of DV to UDLs for a cheaper price than LS, but the implementation is likely to be more complex.

3.4.2 Link State

The Link State (LS) technique was developed later than the DV to address some of its drawbacks. As the name "Link State" stands for, each node maintains a table containing information about each link that exists in the network. In the literature, LS protocols, eg OSPF, are seen as very powerful protocols because routes are computed having a total knowledge of the topology. Keeping a consistent table of LS at each node is done to the cost of complexity. Routes are computed locally, as opposed to DV; therefore this technique adapts very quickly to changes. The decision process is based on a complete knowledge of the topology obtained by flooding, which is the best mechanism for doing so. All nodes therefore have a common point of view of the network one from the other and are able to take the same routing decision, i.e. forwarding a packet on the same best path as all others routers would do. The result has to be always the same, otherwise, routing loops could occur. For doing so, tables have to be kept consistent.

Keeping this consistent view of the network at each node, that is, having the same table at a particular time, is the most challenging topic. This is not realistic because topology changes like failures do occur, consequently topology changes have to be flooded across the entire network as quickly as possible. There are risks of routing loops until changes are propagated everywhere since some nodes might not take the same decision, based on erroneous data (that is the problem of transitory state). The solution is to freeze table entries affected by changes while flooding those, and to acknowledge all the routing traffic hop-by-hop.

LS have limitations in the face of large networks since memory requirements and computational complexity are growing exponentially with the expanding network. One might say that memory and computation power are getting cheaper leaving limitations to a small consideration, but the network is growing quicker.

LS could easily support several metrics if required by adding an additional column for each metric in the table and incorporating information about those metrics in the messages exchanged to establish the map. Applications should then specify the metric they are wiling to use.

Keeping the full knowledge of the network in table provides the ability for multi-path routing. All possible routes could be computed. Likewise, a new path can be computed very quickly when a failure occurs. Doing so doesn't require any further information, but care should be taken that all nodes will compute the same new route. All routers must therefore agree on common policies.

It should be easy in theory to support UDLs with the LS technique since routing tables actually record directed links from one node to an other. There should logically be two records in the table for each bidirectional link. However, LS implementations require BDLs for reply and acknowledgment messages. As this was seen in section 3.3, acknowledgments will have to follow a longer path. This is even not possible at start up to send acknowledgments because the path is unknown. Moreover, obtaining the knowledge or failure of the successors (i.e. directed link that go from the local node to somewhere) is far most difficult in UDLs networks, as seen in section 3.3 than obtaining those coming from predecessors. LS is therefore not applicable in an UDL network.

3.4.3 Path Vectors

We present path vectors (PV) through the characteristics of Border Gateway Protocol version 4 (BGP-4) because this technique has mainly been implemented in this protocol. We will nevertheless temperate strong relevance to BGP-4. PV were developed to address some of the problems of DV and LS, and is an adaptation of the DV technique. In LS, too big tables were feared, whereas "counting to infinity" was criticized in DV, not to say the risk of routing loops in both techniques.

There is no mechanism similar to "counting to infinity" in PV. Nodes maintain a list of nodes traversed for each route. This prevents loops. Moreover, BGP nodes only advertise those routes that they themselves use. A decision process running independently at each node is responsible for the selection of routes which are going to be advertised to its peers. The decision process computes a degree of preference for each newly received route. Routes with the highest degree of preference for each distinct destination are selected and disseminated to peers, which acknowledge their successful reception.

Flooding or exchange at regular intervals of the entire table is not required as it is for LS or DV protocols. Flooding is only necessary initially, and updates concerning particular routes are sent upon routing table changes. Invalid routes are advertised and removed from the tables. The protocol then is looking for a replacement route.

The current specification of BGP requires that traffic sent take the same route in the two directions. This has limitations in face of UDLs since the reverse path is likely different but there is no trouble using different paths since BGP basically allows to administratively set preferred routes. This static setting doesn't clearly address the question of routing in UDL networks. BGP requires acknowledgment of routing traffic as well.

It therefore looks like the PV technique doesn't allow to do dynamic routing in an UDL network.

3.4.4 Current techniques are not applicable

It is hard to objectively pretend that one technique is better than the others, but it clearly appears that current techniques are not applicable to an UDL network, because of the characteristics depicted by UDLs. The flooding nature of LS would be applicable if the acknowledgment of routing traffic was not necessary; on the other hand, the nature of DV (and PV) is highly jeopardized and should likely be very costly in order to provide the same information about distances.

We are therefore constrained to design a new protocol based on a different technique. Both DV and LS can provide bases, but the new protocol will not be classified as DV or LS protocol.

3.5 What are the expectations of a new protocol ?

Because we want to address the most general solution, we will consider a pure UDL network, that is, there is never a direct reverse path between two nodes. Solving this general case ensure that our protocol will work in any situation, even if it might not be very optimal if there are numerous true BDLs, a BDL being simply two UDLs. To what limit our All-UDLs solution would add overhead in this case deserves further study but we can already think of a protocol detecting BDLs and behaving differently in order to avoid overhead.

Another reason for doing so is that solving unidirectional routing as well solve asymmetric routing. Because of the specific characteristics depicted by UDLs, we are indeed dealing with a fundamental change of architecture.

We won't consider the question of multi-cast routing, neither the one of network undergoing rapid reconfiguration, like mobile networks. This is left out for further work also.

Before designing a new protocol, we need to know what knowledge of the topology we want. Having a full and accurate knowledge of the topology is important in order to compute the shortest paths according to various techniques. A full knowledge provides for multi-path routing and secondary paths in face of failure, but we have seen in section 3.3 that this won't be as easy as in UDL networks. We at least need to establish the neighborhood of a node, i.e. predecessors and successors of a link. We already know that establishing a list of successors will be harder than for the list of predecessors. We need as well a manner to maintain this connectivity, therefore a manner to forward data informing of the failure of a successor back to its predecessor.

In order to do optimum routing, we need to know who is reachable, but it is not necessary to know all possible destinations to start routing. If we want to know all possible destinations, we need to ensure we have reached this target by using a termination detection algorithm, which will be more costly in an UDL network.

If the topology is highly dynamic, it will be more difficult for nodes to adapt quickly to changes because the reverse paths might be longer. We need to avoid unnecessary updates. A node is not aware of the failure of the successor as quickly as if it was a predecessor. We should rather look for a protocol that is capable of optimum routing without the full knowledge of the topology.

This lead to the next question: we already know we do not want a full knowledge of the topology, but do we need to keep all possible routes for a particular node, or only the best ?

When the network is getting very large, the amount of routing traffic is increasing exponentially and is consuming too much bandwidth. We face a risk of routing table explosion, but this question, together with the computational capability consumption is seen as less important than the consumption of bandwidth.

As it is said in [25], most of the traffic is intended for destinations in a nearby area. Having a full knowledge of the topology is therefore useless in 80 % of the routing demand, this leaving us with the question of how much knowledge do we need to compute good routes. Basically, we need to have enough knowledge to compute optimal routes in the nearby area.

We could rely on non-deterministic and heuristics methods for those long way away destinations that would help to find efficient routes with a partial knowledge of the topology the first time this route is taken, and trying to improve them if a particular destination is heavily used.

There are various solutions that could apply to large or highly dynamic networks where flooding for instance would be inappropriate because it requires to much time and bandwidth. For instance, on demand routing, where a path is found only when required. That could decrease the amount of information maintained about networks located a long way away but would lead to higher delays.

On demand routing could be implemented using the technique called "circuit detection" as described in section 3.3. This might well avoid routers sending loads of reachability information, which is of prior interest when links are UDLs as seen in section 3.3.

Chapter 4

A Circuit Discovery Unidirectional Link Routing Protocol

In this chapter, we present the proposal of a UniDirectional Link Routing Protocol (UDLRP). First, we present the motivations that drove to the design of this protocol, which is inspired to a certain extent from DV, PV and LS.

Relying on several observations, we will start by discussing in section 4.1 the *design issues* of our protocol that we call "Circuit Discovery UDLRP", based on the detection of circuits, as it was introduced in section 3.3. We first give an *outline of our proposal* in section 4.2 before presenting more informal details about the circuit and topology discovery and the decision process of our *Circuit Discovery UDLRP* in 4.3.

In order to use our protocol in large networks, we need a manner to decrease the length of circuits and the number of feasible circuits a node can see. We found some ideas which help in enhancing the *decision process* for dynamically cutting out the network into pieces. This is next presented in section 4.4, but deserve further research.

A discussion in section 4.6 around *advantages* of our proposal concludes with this chapter and follows section 4.5 that gives a *rough estimate* of the quantity of traffic generated by this protocol.

4.1 Design Issues

The design of the following prototype relies upon several observations. First, the architecture depicted by unidirectional links exhibit some specific characteristics. A network where all links are unidirectional is represented by a directed graph.

Second, because links are oriented, the knowledge of some links is useless to the point of view of some nodes. As an instance, I cannot do anything with the knowledge of the link DA in our simple All-UDL test-bed (figure 4.1). Basically, a node only need to compute and advertise paths it actually uses. No path used by I are going to transit by DA. If we could totally order each nodes, we would see that the order of D is higher than the order of I, which is itself higher than D. Knowledge of links going directly from a higher order to a lower order than a specific node are unnecessary for this node.

Third, most of the protocols currently in use maintain databases in which they record information about routes from the information they received from neighbors. This usually provide information about routes which initiated somewhere else and are ending at the local node. We cannot draw any conclusions from this such like symmetry of routes in UDLs networks. We cannot transpose "You can reach me, with this cost" to "I can reach you with this cost". So we need information about routes that are originating at the local node. But, for instance, collecting the information about the length of the paths that are originating at the local node back to it has a cost, as seen for DV in section 3.4.1.

Fourth, having the local node originating requests and collecting data back to it means that this traffic is coming via another link, possibly another path if most of the links in the network are unidirectional. Basically, traffic has followed a circuit in the network. We are looking for a manner to discover such circuits.

Fifth, whatever the architecture, we always need to establish a bilateral communication between nodes. As seen in a previous paragraph, the difficult part was to obtain the distance leading from the local node to a remote node. We bypass this problem by saying that finding the best path to get to a remote node is equivalent to find the best circuit that goes to this node and come back. The best circuit would simply be the merge between the two best paths in both directions. We are therefore looking for the best circuit where the destination is the source. This greatly simplifies routing when a significantly percentage of links are unidirectional, as seen in DV where it was necessary to provide the information about path length back to the local node.

Sixth, because of the unidirectional nature of a link, it is said that a path exists between two nodes only and only if there is a path connecting the two nodes in both directions, i.e. the graph representing the network is connex. In others words, a remote node is reachable from the local node if this remote node is located on a circuit originated from the local node that passes through the remote node and is ending at the local node.

Seventh, since links are UDLs and that we cannot conclude the symmetry of paths, we should nevertheless be able to use the mere fact that some routing information came from somewhere.

4.2 Outline of the proposal

Our proposal is based on the detection of circuits as presented in section 3.3. Once a circuit is discovered, it is possible to establish a bilateral communication with all nodes that appear on it. Thus, potential destinations are nodes that appear on circuits. The aim of the protocol is to maintain tables containing at each node possible and useful circuits. The protocol doesn't necessarily compute circuits for each destination.

At start up of the network circuits are obtained by a circuit discovery request (CDR) that is forwarded by each node to all their successors. The CDR records nodes on the way and the total length according to various metrics from the departure place to the current node. A decision process running at each node has the responsibility to compute or decide which circuits are recorded in the local circuit table from the information contained in the CDR. It also decides what content is forwarded in the CDR in order to avoid redundancy and computes a degree of preference for each circuit recorded in the table.

The CDR is periodically initiated in order to ensure that optimal routes are found since the topology of the network is dynamic.

All nodes are requested to send periodic KeepAlive Messages (KAM) to their successors. Successors are therefore aware of the failure of their predecessors and incoming links. Successors have the responsibility to inform all nodes that appear on circuits passing through the failing links, using the "selective flooding" mechanism which only propagates notice of failures to nodes on the circuits. Those circuits are put in *disabled* state until they recover.

When a link recovers from a failure, the same mechanism is used to label *enabled* the circuits involved, unless the timer has expired, in which case circuits are deleted from the table.

If this is the case, the procedure is the same as for the addition of a new link. New links are noticed by their successors which inform their predecessors and the network about the existence of a new portion of path. In case of addition of a new node, the process is the same but the predecessors of the new node provide it their circuits.

4.3 A UDLR Protocol Based on Circuit Discovery

For the comprehension of the reader, we use in this section instances on our test-bed network (see figure 4.1 with B as the local node if it is not otherwise specified.

The Circuit Table Each node maintains a table containing possible and useful circuits. Possible and useful circuits are defined by the decision process. The decision process computes circuits and records them in the table from information about circuits obtained by the *circuit discovery request* (CDR) and information already recorded in the table. Ideally, at each node all possible destinations in the network should appear on at least one circuit, possibly more, but this is not a requirement, depending on the size of the network. We could limit the knowledge of circuits in a nearby area. Keeping several circuits at each node allows to provide quickly a substitution circuit in the face of failure, to provide for multi-path routing and to provide for routing according to various metrics. For the point of view of a particular node, possible destinations are the nodes contained in the circuits recorded in the table. The table therefore contains circuits entries. For each circuit, we record an identification number unique through the network, the total length according to various metrics, likely a degree of preference and a label specifying the state of the circuits (*possible, selected, enabled, disabled*). The table should appear as a list of destinations indexed by the circuits they are appearing on.

4.3.1 Determining Neighbor Reachability

The purpose of the neighbor-reachability mechanism is to confirm that the neighbors can safely be considered operational and capable of providing reliable reachability information. Nodes are therefore keeping an up-to-date list of all theirs predecessors and successors by demonstrating their own operationality. Successors of links involved in changes are responsible for the proper delivery of notification to all relevant nodes in the network.

KeepAlive Messages The identity and operationality of predecessors is ensured by KAMs sent on each outgoing links, while successors are considered operational unless their failure is explicitly notified. This is therefore the responsibility of the successors of failing links to give the failure notice to the network. KAMs should be sent randomly to avoid unexpected synchronization of nodes, this leading to a periodic overflow of control messages in the network. A node sends KAMs as long as it receives KAMs from at least one of its own successors. If this is not the case anymore, the node is not reachable and there is no possible substitution circuit. A node in this situation doesn't transmit data anymore and puts itself in the state *disabled*. It as well stops sending KAMs until it receives KAMs from one of its predecessors. Listening to KAMs provides the knowledge of the predecessors, while receiving circuits originated at the local node ensures that the first hop on the circuit is one of its reachable successor. In addition, receiving those circuits back to the local node ensures that predecessors are reachable.

Selective Flooding We will often refer to the *selective flooding* mechanism in the following sections. Its purpose is to ensure circuit consistency, which is limited from circuit to circuit. The same vision of a particular circuit has to be maintained at each circuit member. This process is very similar to flooding, but instead of forwarding a message to all successors, it only forwards it to a subset of those which appear on circuits passing through the initiator of the message. This is easily done at each node by forwarding the message to successors that appear on circuits passing through the initiator and themselves. If all possible destinations in the network appear on circuits at the initiator, all the network will be flooded, otherwise only relevant nodes. The cost of this mechanism is therefore cheaper or equal to flooding.

Failure Notifications Upon detection of a failure, the immediate successors sends a notification of failure to their successors using the *selective flooding* mechanism. All nodes on the way mark relevant circuits as *disabled* and set a timer. Its expiration guarantees that circuits are permanently deleted from the table if links don't recover from their failure after a specified amount of time. With this mechanism, all nodes which have circuits in their tables passing through the failed link are aware of the failure, amongst them the predecessors of the link out of order. This doesn't prevent them to keep sending KAMs. If the last incoming link of a node is out of order, the node is aware that it is unreachable and that there is no way to find a substitution circuit. It stops sending KAMs and put itself in an out of order state until it receives KAMs again from any predecessor. This snowball to all successors that have only an incoming links using *selective flooding*.

Recovery after failures Once links have recovered from their failure, the immediate successors again receive KAMs from predecessors which were keeping sending those to allow successors to be aware of the recovery. Obviously, there wouldn't be KAMs if predecessors are not themselves operational. *Selective flooding* is used since entries of circuits involved were kept in the tables. Relevant circuits are set to *enabled.* If the timer has expired, the recovery is then equivalent to the addition of a new link.

Addition of a new link In growing networks, we cannot afford to proceed to the same circuit discovery each time a new link or node is connected to the network as it is done periodically or when starting from scratch. This would be the easier way, but this adds too much overhead. Note that connecting a new node to the network necessarily adds new links. Basically, end-points of new links absolutely need to be aware of the new connections, otherwise those would be useless. Moreover, all nodes in the network might compute new shortest circuits, and an incoming node needs to fill its full table. First, KAMs are immediately sent on new links. Successors can establish new circuits from this information and what is already recorded in their table. Then, successors are advertising the new portion of path and its metrics to all nodes that have circuits passing through the predecessor and the successor of the new link using *selective flooding.* All relevant nodes add their name and costs and can update their tables to add new computed circuits in the table from the information they already have. As an instance in our figure 4.1, the addition of the link DA in the network would be notified to A by KAMs received directly from D. A advertises the new portion DA, so that B can deduce new circuits to reach A. If it already knows the circuit $BCDIAB$ it would deduce $BCDAB$. If B doesn't have the knowledge of D or A, B doesn't get the notice of DA until a new CDR is initiated and finds DA, which depends of the decision process. In any case, the predecessor has to be informed of the new link. Without any feedback, the node will know it is not connected yet. Obviously, this wouldn't provide all best circuits until the periodic CDR is performed again. If we urge to do so, we can improve the result by flooding the new portion of path to the entire network. Some nodes might not have the possibility to use this information, but they have the knowledge that the new link exists. In the case of addition of a new node, its predecessors, once aware of it, send it their circuits. The rule, for the point of view of a new node, is that all its circuits are necessarily passing through its predecessors. Here again, we could inform the network of this new node by flooding. For instance, if we were adding the node X between D and A, X is waiting for KAMs from D, and then sends KAMs to A and advertises DA. A then transmits DXA, and D, at the reception of this messages, knows it has the send all its circuits to X. Here again, our solution doesn't cost more than flooding.

4.3.2 Circuit Discovery

This consist in providing information about circuits to nodes when the network is switched on and at periodic times. The relevant period of time is not clear yet and will require more consideration.

The Circuit Discovery Request The CDR is initiated at some node and is forwarded by nodes on the way to all their successors. Nodes on the way are recorded in the current request together with some information about total length of the path from the originator to the local node according to various metrics. Basically, each current request is following a different path and is recording different information. Ultimately, some requests are back to the initiator and are ended there. At each node, a *decision process* is responsible to extract what is useful to compute new circuits at the local node. It also decides which requests and what content should be forwarded to its successors in order to avoid redundancy and infinite loops. The process of circuit discovery could in theory be initiated by all nodes of the network. Although quicker, that would add too much overhead and redundancy. Indeed, the same information about a given circuit would be collected as many times as there are nodes in the circuit, but with a distinct initiator at each time. On the other hand, finding a single initiator might require an election mechanism. This mechanism might well add too much overhead just for the purpose of obtaining the dedicated router. If finding a dedicated router is easy, we recommend to have a single initiator, otherwise the rule would be to stop CDR if they don't provide new sequences of nodes. But this case is already taken into account by the decision process, therefore our protocol is able to work in both situations, at the discretion of the administrator.

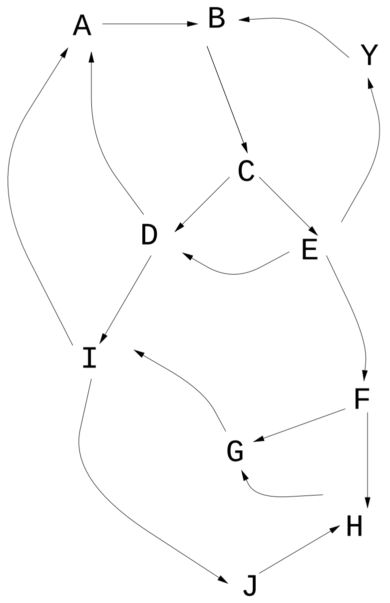


Figure 4.1: Demonstration Network

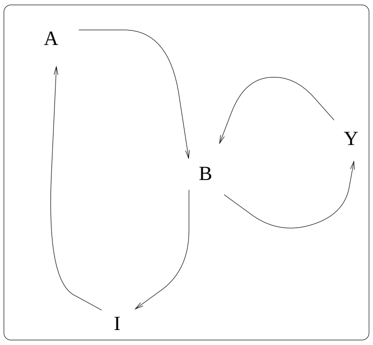
The Decision Process At the reception of a CDR, the local node is passing the information contained in the message straight to the decision process. The decision process is responsible for computing local circuits, to record useful information obtained from the CDR and to decide what circuits are useful and what information is forwarded to its successors. Before forwarding the CDR, the decision process checks if it hasn't already sent the same sequence of nodes. The decision process then computes a degree of preference for each of those circuits it accepts in the table and labels the best ones *selected*. The decision process checks the CDR in order to know if it was the initiator of the request or if the CDR already contains a circuit.

The CDR was initiated by the local node In this case, the first node of the sequence of nodes is the local node and the sequence represents a circuit (eg: $BCDA$) [1]. The decision process records in the table and labels it *possible* and there is no need to further forward this CDR.

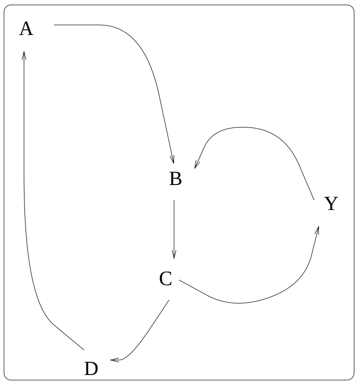
The CDR was initiated somewhere else If it wasn't the initiator, it checks if the name of the local node already appears in the list of nodes. For instance, B could receive $BCEY$, IA, or $ABCEY$. If its name doesn't appear, this list of nodes is not a circuit, but a path for the point of view of B. It is recorded in the table with the status *incomplete circuit* for later use by the decision process if the same sequence doesn't already exist in the table. If this sequence is a new one, the local node adds its name at the end of the sequence and its metrics to the total lengths, and then forwards it to its successors. We record *incomplete circuits* because new circuits could be computed from the incomplete ones. We keep relevant data that might help to boost the circuit discovery process. Another aim is to avoid forwarding several times the same information. If the local node does already appear in the sequence (eg $ABCEY$), a circuit initiating at the local node has been found, but the decision process has to check its admissibility in the table.

Extraction of circuits contained in the CDR If the local node does only appear once on the sequence (eg: $ABCEY$), a portion of the sequence is a circuit for the local node ($BCEY$ in the same instance). The decision process extracts it and adds it in the table with the status *possible* and forwards it to its successors after the addition of the local node at the end of the sequence and its metrics to the total lengths. Note that the sequence $ABCDA$ cannot be received at B since A wouldn't have

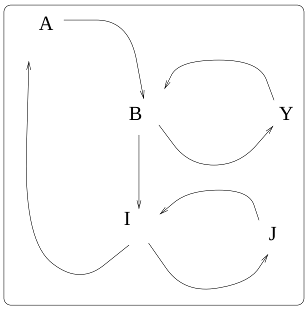
[1]Note that the local node B is not recorded at the end of the sequence since it was just received as is from the predecessor. This convention will always be used



Circuit ABYBIA allowed to join'Y' from 'A'



Circuit 'ABCYBCDA' allowed to join 'Y' from 'A'



Circuit ABYBIJIA forbidden to join 'Y' from 'A'

Figure 4.2: Allowed Circuits

forwarded it. On the other hand, the instance $IABCDA$ where A appears two times is perfectly right since A was in the situation presently discussed. Moreover, for the point of view of B, this sequence has to be transmitted to I to teach it the reachability of D for instance, even if I might eventually discard it. This has to be done because B doesn't know if I can learn the existence of D by some other path, neither do it know which is the shortest path. If it appears two times on it, the decision process drops it off since it cannot be an optimal path. The information may nevertheless be useful. For instance, at the reception of $IABCDABCEY$, B would forward $IABCEY$ so that I is aware of Y. The part $IABCDAB$ was already forwarded before, so Y was already ensured to obtain the knowledge of D. On the other hand, nothing is forwarded at the reception of $ABCEYBCEY$ since $BCEY$ was already extracted and $ABCEY$ forwarded previously. Basically, the decision process checks if new nodes appear since last round and only forward the relevant part of the sequence.

Another instance To clarify, here follows a second instance at node D. The decision process would drop off $IABCDABC$ because A, B and C were already recorded. On the other hand, it wouldn't drop off $IABCDABCE$ since E is new, but it would forward only $IABCE$. Note that at the previous passage D forwarded $IABC$. I will therefore be aware of circuits $IABCD$, $IABCED$.

Sub-circuits As the reader might have noticed, some sub-circuits might appear in a circuit, as for instance $BCEYB$ in $ABCEYBCDA$. This could be seen as a routing loop, but this knowledge is perfectly right in order to reach Y. The data packets will necessarily have to pass two times by node B. On the other hand, there should never exist two sub-circuits in a circuit. Our decision process is therefore checking this particular case. For instance, there are two sub-circuits $BCEYB$ and $IJHI$ in the circuit $ABCEYBCDIJHGIA$. No optimal circuit is using this, but the knowledge of $IJGI$ is useful to reach J and $BCEYB$ to reach Y. Therefore, when the decision process at node I receives $ABCEYBCDIJHG$, it doesn't forward $BCEYB$ because it has already forwarded this last time it received $ABCEYBCD$, but it forwards $ABCDIJHI$ to its successors, including J since I doesn't necessarily knows if there is a path to reach A by J. Figure 4.2 summarizes which circuits could be allowed.

Validation of Selected Circuits Before being fitted for routing, selected circuits have to be validated in order to ensure consistency amongst their members, since we cannot ensure that all nodes made the same decision. The validation is initiated by the node on a particular circuit with the smallest number, that we call the circuit origin. The origin then sends messages containing circuits it wants to check directly to the next hop on the circuit. Each node forwards it to the next hop of the circuit until it comes back to the origin. All nodes on the way record the circuit in the table and set a timer. After its expiration, the circuit is labeled *enabled* unless a failure notification is given. When it gets back to the origin, it knows that all nodes on the circuit are aware of it. It therefore sets it to *enabled* as well. It it doesn't come back after a specified amount of time, the origin starts the validation again, or it sends a failure notification to the circuit members. Data packets can be sent to all destinations on {it enabled circuits.

4.4 Enhancements of the Decision Process

Various reasons might decide for the non selection of circuits. We have implicitly outlined some of the heuristics used in the previous sections, but the decision process will use some more in order to drop unnecessary circuits. Otherwise, there are to many possible circuits, as it is shown on our sample network. We need to find heuristics in order to diminish the size, the number and the cost of circuits when the network is significantly big.

Particularly, we are considering nodes or portions of circuit that are *inevitable* gateways in order to reach some destinations. Nodes willing to access destinations beyond those those nodes don't require the knowledge of the complete path. From their point of view, all nodes beyond inevitable nodes can be seen as direct neighbors of an inevitable node. It will be its responsibility correctly route packets beyond this point. For instance, we have to pass through B if we want to reach Y. Our idea is therefore to dynamically discover those *inevitable* nodes. We are looking for algorithms that discover set of nodes that are connex in the oriented graph area. [4, 3, 5] should provide our first bases.

The decision process could also select circuits according to heuristics based for instance on the distance of the circuit compare to the diameter of the network. Generally speaking, it uses information labeled "incomplete circuits" for doing so. We decide to record this information because its knowledge is very useful and the cost to maintain it not very important.

4.5 A rough estimate of the number of possible circuits

In this section, we provide a method that give a rough estimate of the number of messages which can be deducted from the number and length of circuits. This basically gives the upper limit of messages, because we believe that our decision process can perform better once we have found better heuristics.

We first determine the length x in hop-count of the longer possible circuit, and use it to compute G^x, where G is the matrix representing all links in the network. The maximum length of the longer circuit in a network of z nodes is theory $(z-1)*2$.

The element (i,j) of G^x is the number of paths from i to j of length x. By extension, (i,i) is the number of circuits from i to i. In others words, values on the diagonal of G^x represent the number of circuits of length x. Then, we only use values on the diagonal. There are no values on the diagonal of G since there are no circuits of length 1. We can therefore check is there exist a circuit of length x.

The maximum number of circuits to go from i to i is $\sum_{l=1}^{x}(i,i) \in G^x$.

There are obvious redundancies in this count. For instance, G^8 records the number of circuits of length 8, and only 8. It therefore records two times $ABCDA$ which has length 4. The total number of circuits that pass by a specific node is therefore the total of circuits of each length, minus those we count twice. Moreover, each circuits of length x is counted x times. We therefore need to subtract $(x-1)$ circuits of length x. This gives the number of circuits we can discover and could give an idea of the number of messages required to obtain all circuits. From the reminding total of circuits, we could subtract more according to the heuristics.

In our test-bed, the longer possible circuit has length 16 while the longer selected circuit has length 13. It allows the communication between 'Y' and 'J'. In theory, we should compute $\sum_{l=2}^{16} G^l$ to obtain all possible circuits.

As an instance, we limit ourselves to the knowledge of circuits passing through A. Node A can reach all nodes in at most 9 hops, therefore we simplify the computation and we only show the number of circuits of length 9.

First, here is the matrix accounting the number of possible paths of length 9.

$$\begin{pmatrix} A \\ B \\ C \\ E \\ E \\ F \\ G \\ H \\ I \\ J \\ Y \end{pmatrix} : G^9 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^9 = \begin{pmatrix} 7 & 5 & 1 & 3 & 1 & 2 & 3 & 3 & 5 & 3 & 2 \\ 8 & 9 & 5 & 2 & 1 & 1 & 5 & 5 & 6 & 5 & 1 \\ 8 & 9 & 9 & 6 & 5 & 1 & 6 & 6 & 7 & 6 & 1 \\ 4 & 2 & 3 & 7 & 5 & 2 & 2 & 2 & 5 & 2 & 2 \\ 9 & 7 & 6 & 7 & 4 & 3 & 5 & 5 & 7 & 5 & 3 \\ 3 & 5 & 3 & 0 & 0 & 0 & 3 & 3 & 2 & 2 & 0 \\ 1 & 1 & 2 & 3 & 3 & 0 & 1 & 1 & 2 & 1 & 0 \\ 1 & 2 & 3 & 0 & 0 & 0 & 2 & 1 & 1 & 1 & 0 \\ 5 & 1 & 1 & 5 & 2 & 3 & 1 & 1 & 4 & 2 & 3 \\ 2 & 3 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 1 & 0 \\ 7 & 5 & 1 & 3 & 1 & 2 & 3 & 3 & 5 & 3 & 2 \end{pmatrix}$$

We would have 15 circuits passing through A in the total matrix from length 2 to 9, scattered like that:

Length	Number
1	none
2	none
3	none
4	1
5	2
6	1
7	1
8	3
9	7
Total	15

Those marked NO wouldn't have been propagated by the decision process.

	Circuit	Valid
1	ABCDA	OK
2	ABCDIA	OK
3	ABCEDA	OK
4	ABCEDIA	OK
5	ABCEYBCDA	OK
6	ABCEYBCDIA	OK
7	ABCEYBCEDA	OK
8	ABCEFGIA	OK
9	ABCEFHGIA	OK
10	ABCDIJHGIA	OK
11	ABCDABCDA	NO
12	ABCDABCDIA	NO
13	ABCDABCEDA	NO
14	ABCDIABCDA	NO
15	ABCEDABCDA	NO

4.6 Advantages of our approach

We have presented in this chapter an UDLRP based on circuit discovery and inspired from existing techniques and their behavior. Although this protocol is at the level of current study and has therefore not been validated, we can already exhibit some of its expected advantages.

First, our approach allow a quick adaptation to failures by providing for multi-path routing. The protocol quickly informs the network about such failures without deleting circuits involved. It as well allows routing according to various metrics.

Second, there is no need for explicitly maintaining coherent tables. This architecture doesn't exhibit the periodic exchange of tables in order to do so, but it rather uses the circuit discovery request which is less costly in order to find more optimal circuits than ensure consistency. Routing is done accurately if all nodes that appear on a circuit have the same vision of this circuit. The consistency is therefore limited from circuit to circuit.

Third, there is no need to know the cost to get to a specific destination, as seen in standard techniques. The shortest circuit passing via the node under consideration is the shortest path to go to this destination plus the shortest to come back. At the local node, we just need to know the cost of the circuit in order to find out which one is the shortest.

Fourth, our protocol doesn't require a full knowledge of the topology, neither the knowledge of all destinations. It as well allow a partial use of the network since destinations that appear on *enabled circuits* are reachable even if all circuits are not found yet. Moreover, it (should) provides a manner to dynamically divide the network in order to diminish size of tables and size of circuits. Moreover, the decision process is based on various heuristics for this purpose.

Fifth, more heuristics can easily be implemented in the decision process in order to further diminish the size of the circuits. Our protocol seems well adapted under the assumption that the return path is taking a route highly different to the one taken for the outward journey, i.e. if an important portion of the links are not actually bidirectional. The hypothesis that the traffic is taking another way was one of our observations.

Sixth, because it can limit its knowledge to a limited nearby area, topology updates are cheaper than flooding to the entire network, although the concept is very similar.

Seventh, circuits don't have to be computed every time a packet is received. Circuits are recorded and just have to be found in the table. Routing should be quicker with this technique.

Chapter 5

Open Issues

In this chapter, we present open issues which deserve further studies in order to enhance what we have presented in this report. Moreover, we briefly introduce some ideas which could be useful.

Omni-directional Links Our proposed Circuit Detection UDLR protocol doesn't take into account the type of UDL. Basically, this could deserve importance, since a satellite link, by nature, is omni-directional. It, in theory, is able to reach any destination. Further study might therefore specify a link as being omni-directional.

Comparison of Current Techniques In order to enhance the arguments to the applicability of LS, DV and PV, we should consider several points. First, we should compare those in term of number of messages and bandwidth consumptions according to various situations, like a pure BDL network, a pure UDL network, a mix of BDL and UDL and finally a pure BDL network where each BDL is replaced by one UDL in each direction. We should compare and analyze their performances. Second, we should exhibit in which of UDL and BDL networks the risk of routing loop is higher. We think the risk is limited in an All-UDL network.

Dynamic networks Our study doesn't consider networks undergoing rapid configuration since satellite UDLs might be of great interest for mobile networks (without need to reallocate new addresses every time a host is moving as soon as it remains under the footprint of the satellite). Moreover, the foreseen use of low-earth orbit satellites are likely to take benefit of this enhanced research.

Optimum Routing We have focused on networks where all links are UDLs. Because many links might remain BDLs in the long term, we need to take them explicitly into consideration in order to avoid overhead. A mechanism that detects BDLs is easy. In our Circuit Detection UDLRP, this would require to detect nodes sequences such like XYX, i.e. circuits which only discover one node. The interesting question is how to allow their cohabitation in an optimum manner.

Knowledge of the Topology In order to establish the topology, we could design a recursive algorithm which requires the knowledge of the predecessors which themselves require the knowledge of their predecessors, on so on until it discovers the successors, or the opposite way round. Because our proposition doesn't require a full knowledge of the topology, we should define a way to compute degree of preference of circuits based on statistics for instance. That could improve their right selection.

Large Networks In order to diminish the size and number of circuits, we are looking for a way to automatically detect connex components in the Internet, that is, detecting BDLs islands.

Architecture of the network We have a problem for instance if there are many paths, say 500, to get from A to B. All 500 paths would be forwarded by B on BA. This is nevertheless not relevant to our UDLR protocol, but to the architecture of the network.

Decision Process Our decision process is not designed yet. It requires more consideration before an implementation attempt is made possible.

Others domains of consideration Our proposal doesn't consider the foreseen enhancements that will be soon given to the Internet such like Ipv6, particularly the addressing politic that is going to be used. This might require serious updates to our proposed protocol since the problem of long circuits might be reconsidered.

Circuit Discovery Request We don't know yet what is the adequate periodicity of this request. This brings importance in the comparison between protocols performance. The longer the period is, the better our proposition is.

Theory of graphs We should gain important improvements by studying relevant papers published in the domain of "graph theory" since All-UDLs networks merge with problems of directed graph. We should nevertheless note that the theory of graphs has found very interesting solutions in graphs where the topology is known and static. This is not the case in the Internet. This brings serious limitations to the use of directed graphs results, but we will nevertheless take a careful look on it.

Chapter 6

General Conclusion

In this report, we have considered the problem posed by UDLs to routing. Our study considered three different architectures where UDLs could exist.

The first one is depicted by an alternative satellite UDL on top of a BDL network. It requires a back-channel in order to transmit routing information about subnetworks back to the feed. This case is currently under investigation by the working group UDLR at the IETF. The working group has given birth to two approaches. Tunneling which set a tunnel over the back-channel to emulate a satellite BDL and consequently hide the actual topology to the routing protocols, and RPM which explicitly takes into account the unidirectional nature of the satellite link in the routing protocols. Our comparison has shown that the same problems need to be resolved by the two approaches, in particular the question of inter-domain routing which was not addressed at all, an for which we made suggestions. Our discussion on both approaches provided material for the working group to agree on tunneling for the short term.

The second case represents islands made of BDLs that are interconnected by UDLs, while the third architecture is the general case where all links are UDLs in the network. Both architectures are indeed very similar since BDLs islands could simply be seen as nodes, which translates in the general case and is raising the same questions. Because this is rather seen as long term, it is not taken into account at the IETF which only focuses on the short term.

We have therefore considered both cases together and we have reached the conclusion that a new routing protocol should be designed since solutions for the short term are not applicable. To reach this conclusion, we have exhibited the characteristics of UDLs and we have studied current techniques in use in the Internet, namely Link State, Distance Vectors, and in some extent Path Vectors. UDLs have impact on the way distances are computed, acknowledgments processed and the topology and neighborhood are discovered. Distance Vectors showed serious limitations because distanced cannot be provided while Link State are limited by the processing of hop-by-hop acknowledgments.

We have therefore proposed a protocol for a network entirely made of UDLs because this simplifies the study. The interesting side effect is that solving UDL in an All-UDL network allows at the same time to solve the question of link asymmetry. A BDL is simply two UDLs, one in each direction, with likely distinct metrics.

Our proposition is called "Circuit Discovery Unidirectional Link Routing Protocol" and was described in informal details and is inspired in some extent from current existing techniques. It requires more investigations and we are currently working on it in order to propose a full specification. As its name stands for, its aims is to discovers circuits, i.e. paths that come back to the departure place, based on the fact that a node is reachable only if there exists a path in the two directions. This technique doesn't require neither hop-by-hop acknowledgment processing nor computing distances to destinations.

The protocol provides for multi-path routing, and doesn't require a full knowledge of the topology. It doesn't necessarily compute all possible circuits; those might even been computed on demand only. It is not proven yet that our protocol ensures consistency of the tables, but it doesn't prevent it to work with suboptimal circuits in the case of transitory areas. We are anyway working on it. There is therefore scope for heuristics techniques in the face of large networks. Moreover, our protocol has not yet been validated nor tested and implemented, neither the decision process been designed.

We rather aimed to provide foundations and propositions to the question of UDL networks. That's the reason why we tried to keep the things simple, that is, we only focused on a pure UDL network and we didn't consider multi-cast routing neither explicitly IGP and EGP.

Glossary and Abbreviations

UDL : UniDirectional Link.
bf UDL network : A network where there exist UDLs.
bf All-UDL network: A network where all links are UDLs.
BDL: Bidirectional Link.
true BDL: A standard bidirectional link. 'Not true' means that there are indeed two UDLs, one in each direction instead of a BDLs.
bf BDL network: A network where all links are BDL.
UDLR: UniDirectional Link Routing.
RPM: Routing Protocol Modification.
LS protocols: Link State protocols.
DV protocols: Distance Vector protocols.
PV protocols: Path Vector protocols.
multi-homed host: a host with connectivity through two or more interfaces.
predecessor / uplink node: the origin end-point of an UDL.
successor / downlink node: the destination end-point of an UDL.
outgoing link: link going to a successor.
incoming link: link coming from a predecessor.
feed: a station which is emitting data towards a satellite on an UDL.
dish: the station which is receiving data from a satellite on an UDL.
circuit: path leading back to the departure place
AS: Autonomous System - aggregation of networks under the administration of a single routing protocol.
IGP: Interior Gateway Protocol: routing protocol inside an AS.
EGP: Exterior Gateway Protocol: routing protocol between ASs.
Circuit Member : A node that appears on a particular circuit.
BDL islands: entities inside which all nodes can establish a bidirectional communication with any other node without the help of any external link.

Bibliography

[1] UDLR archives. ftp://zenon.inria.fr/rodeo/udlr/archive.txt.

[2] V. Arora, N. Suphasindhu, J.S. Baras, and D. Dillon. Asymetric Internet Access over Satellite-Terrestrial Networks. http://www.isr.umd.edu/ccds/research/aia/sld001.htm.

[3] Jean-Claude Bermond and Jean-Claude Konig. General and Efficient Decentralized Consensus Protocols 2. In M. Cosnard, editor, *Parallel and Distributed Algoritms*. LRI, CNRS, Université Paris-Sud, Elsevier Science Publishers B.V (North-Holland), 1989.

[4] Jean-Claude Bermond and Jean-Claude Konig. Un protocole Distribué pour la 2-connexité. In *Techniques et sciences Informatiques*, volume 10, page 269, 1991.

[5] Jean-Claude Bermond, Jean-Claude Konig, and Raynal Michel. General and Efficient Decentralized Consensus Protocols. In *Lecture notes in Computer Science*, number 312. LRI Informatique, U.A. CNRS, Université Paris-Sud, IRISA, Université de Rennes, 1987.

[6] A. Conta and S. Deering. Generic Packet Tunneling in IPv6. Internet draft <draft-ietf-ipngwg-ipv6-tunnel-07.txt>, December 1996.

[7] Scott M. Corson and Vincent D. Park. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *proceedings of the IEEE Infocom*, Kobe, Japan, April 1997. University of Maryland, USA.

[8] Noritoshi Demizu and Suguru Yamaguchi. DDT - A Versatile Tunneling Technology. In *Proceedings of the INET 94/JENC5*. Nara Institute of Science and technology, 1994.

[9] 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.

[10] 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.

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

[12] 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.

[13] J. J. Garcia-Luna-Aceves. A Unified Approach to Loop-Free Routing Using Distance Vectors or Link States. In *proceedings of the SIGCOMM*, 333 Ravenswood Avenue, Menlo Park, California 94025, 1989. Network Information Systems Center - SRI International. garcia@sri.com.

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

[15] C. Hedrick. Routing Information Protocol - RIP. Request for Comments 1058, Rutgers University, June 1988.

[16] Christian Huitema. *Routing in the Internet.* Prentice Hall, 1995.

[17] Hidetaka Izumiyama and Akihiro Tosaka. Uni-directional 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.

[18] G. Malkin. RIP version 2 - Carrying Additional Information. Request For Comments 1388, Xylogics, Inc, January 1993.

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

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

[21] Venkata Padmanabhan, Hari Balakrishnan, Keith Sklower, Elan Amir, and Randy H. Katz. Networking using Direct Broadcast Satellite. In *Proc. of the 1st international Workshop on Satellite-based Services*, Rye, New-york, November 1996. University of California at Berkeley.

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

[23] Y. Rekhter and P. Gross. Application of the Border Gateway Protocol in the Internet. Request for Comments 1772, T.J. Watson Research Center, IBM Corp and MCI, March 1995.

[24] 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.

[25] Joel M. Snyder. A Routing Architecture for Very Large Networks Undergoing Rapid Reconfiguration. In *proceedings of the SIGCOMM.* The University of Arizona - MIS Department, 1989.

[26] 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.

[27] Andrew S. Tanenbaum. *Computer Networks - Third Edition.* Prentice-Hall International, Inc, 1996.

[28] P. Tsuchiya. Mutual Encapsulation Considered Dangerous. Request for Comments 1326, Bellcore, May 1992.

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

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

[31] R. Woodburn and D. Mills. A Scheme for an Internet Encapsulation Protocol. Request for Comments 1241, SAIC and University of Delaware, July 1991.

[32] Yongguang Zhang and Son Dao. Integrating Direct Broadcast Satellite with Wireless Local Access. In *Proc. of the 1st international Workshop on Satellite-based Services*, Rye, New-york, November 1996. Hugues Research Laboratories.