IMPLEMENTATION OF AN IPV6 MULTIHOMING INGRESS FILTERING COMPATIBILITY MECHANISM USING AUTOMATIC TUNNELS

Carlos Barcenilla, Antonio Tapiador, David Fernández, Omar Walid and Tomás P. de Miguel

Department of Telematics Engineering, Telecom Engineering School (ETSIT), Technical University of Madrid (UPM), Madrid, Spain

Abstract: Nowadays, many organizations need to be multihomed in order to achieve fault tolerant Internet access. Unfortunately, the hierarchical nature of IPv6 addressing architecture poses some threats on multihoming. The IETF is designing a solution based on the discussion of several approaches to solve the problem. Ingress filters are part of the problem, so ingress filtering compatibility mechanisms are needed. This paper discusses the host-centric multihoming approach and describes an implementation of an ingress filtering compatibility mechanism based on automatic tunnels and anycast addresses. The implementation has proven to work properly, being easily developed and deployed.

Keywords: IPv6, multihoming, ingress filtering

1. INTRODUCTION

Internet connectivity has a strategic importance for a growing number of organizations nowadays. Their activity heavily depends on the reliability of their Internet connections, as an important part of their mission-critical processes are based on distributed applications running over Internet. In some cases, it is not exaggerated to state that the organization activity completely stops whenever its Internet connection goes down.

For that reason, reliability is an important requirement in organization's demands to Internet Service Providers (ISP). Apart from the use of highly reliable equipment (carrier class) and redundancy in ISP services, organizations try to improve the reliability of their Internet connectivity by getting service from two or more ISPs.

A site connected to Internet through two or more ISPs is commonly known as a *multihomed site*. Multihoming allows achieving fault tolerance: if the main connection goes down, the routing system reacts and redirects the traffic though one of the alternative connections. Besides, multihoming allows an organization to define its own *traffic engineering* policies, ranging from simple load balancing strategies to more complex approaches.

However, as implemented today, the advantages that multihoming presents for the organizations are transformed in drawbacks when they are observed from the Internet global routing system point of view. Multihoming has an important impact on the size of the global BGP routing tables (each multihomed site contributes with at least one prefix), increasing the global routing instability, as stated in several studies [Bu et al., 2002].

Many proposals have been made to address the multihoming problem in IPv6, but there is little work on implementation and validation of the feasibility of the proposals. The aim of this work was to implement and test one of the proposals we considered interesting. Therefore, we designed an implementation of an ingress filtering compatibility mechanism based on automatic tunnels and anycast addresses.

The article is organized as follows. Section 2 introduces the multihoming problem in IPv6 and the solutions proposed so far. Section 3 presents the basis of the host-centric multihoming approach, detailing the source and destination address selection procedures. Later, Section 4 describes the automatic tunneling proposal based on dynamic routing protocols and anycast addresses. Section 5 provides details about how we implemented and tested the solution over Linux and FreeBSD operating systems. Finally, Section 6 summarizes the main conclusions of the paper.

2. MULTIHOMING IN IPV6

As already mentioned, the goal of multihoming is to provide solutions for the management of multiple Internet connections for a site, in order to achieve fault tolerance and support load balancing. Multihoming in IPv4 is traditionally achieved by means of the use of *provider independent* (PI) address ranges announced through all ISP connections of a multihomed site. For example, if an organization is multihomed to ISPs A and B (Fig. 1), their PI address prefixes are announced to A and B. In this way, in case of a

failure in ISP A, external hosts may still connect to the multihomed site through ISP B, being the BGP interdomain routing responsible to react and provide failure tolerance.

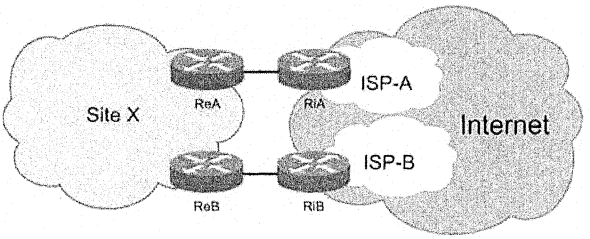

Figure 1. General multihoming scenario.

This technique can also be applied using *provider aggregatable* (PA) addressing. In this case, the site address prefix is assigned by one of the providers, and the other provider agrees to distribute it through its routing system.

In any case, although widely deployed, the use of this IPv4 multihoming technique raises important problems for the global Internet routing system, compromising its scalability.

On the other hand, IPv6, the new Internet network protocol designed by IETF, tries to solve the lack of addresses, widely increasing the address space for the next generation Internet. In addition, it provides multiple features and facilities, such as address auto-configuration, mobility, security or QoS capabilities. In order to improve the network scalability, the IPv6 routing architecture has been redesigned following a strongly hierarchical model. Address aggregation is the key design premise, requiring address delegation from providers to organizations.

Usually, organizations are supposed to be connected to the IPv6 network only in one point in the hierarchy, receiving only one address prefix from its provider. However, as stated before, an important number of organizations need two or more connection points to the network. For example, multinational organizations connected to providers in different countries or, as in the UPM case, university departments connected to production and experimental research networks. In this case, organizations inherit several IPv6 address prefixes, one from each connection.

Due to the strictly hierarchical nature of the IPv6 routing model, IPv4 multihoming solutions are not valid in IPv6 networks, as it is not possible to announce the prefix inherited from one provider through another ISP connection. That behavior would break the address aggregation scheme – 48-bit site prefixes typically assigned to organizations would be flooded into

interdomain routing– and it would lead to unmanageable routing table sizes, especially in core routers (routers in the Internet core that do not have a default route in their routing tables), breaking the whole interdomain routing system. Therefore, in order to keep address aggregation, new solutions to multihoming scenarios are required for the IPv6 deployment.

An IETF working group, Multi6, [multi6] was formed to study the IPv6 multihoming problem and its proposed solutions. Its charter included studying multihoming common practices, defining multihoming goals and requirements, creating a multihoming functional architecture decomposition in IPv6, and defining architectural approaches to IPv6 multihoming solutions. This group has produced an RFC and several drafts, and it has recently been rechartered into a new IETF group, Shim6, which is trying to design a multihoming solution.

Many IPv6 multihoming solutions have been proposed [Tapiador et al., 2004][Dunmore, 2003] . Some of them may be complementary, as multihoming is a complex problem and its solution will probably come from the combination of some of them.

There are site-oriented solutions that propose a global solution to the entire multihomed organization as a whole [Hagino and Snyder, 2001]. On the other hand there are host centric solutions, which are focused in multi-addressed hosts. The most relevant work in this subject [Huitema, 2004] describes the multihoming problem from the host point of view.

Other solutions are based on:

- Network layer modifications, either implemented on the hosts, in the routers or both.
- Transport layer modifications.
- Addition of new layers.

Some of them, like MHAP [Dunmore, 2003], have already been discarded. Solutions like LIN6 [Teraoka et al., 2003], MAST [Crocker, 2003] or HIP [Moskowitz et al., 2005] have been reviewed, and many of their ideas will be gathered in the multihoming Shim6 solution. This host centric approach will insert a new sub-layer (shim) inside the IP layer of the end systems. Packets traversing the network will use the multiple IPv6 available addresses at each host (so called locators) in order to establish new connections or maintain the existing ones after a network outage. But the shim layer will show steady addresses (ULIDs) to the upper layers. In that way, transport connections can survive locator changes, i.e. address changes. Maintaining state at both ends of the communication will be necessary.

To prevent address spoofing attacks, ISPs perform *ingress filtering* on packets leaving the site and entering the ISP's domain. Filters check that each packet's source address belongs to the prefix delegated to the site by

the ISP. Packets not passing the check are dropped by the ISP's access router.

Among other issues, the Shim6 working group will have to work on solutions to avoid ISP ingress filters. These solutions for the ingress filtering problem are commonly called *ingress filtering compatibility mechanisms*.

The rest of this paper will deal with in the Host Centric multihoming proposal in a general manner, and with an ingress filtering compatibility mechanism we have implemented in a specific manner.

3. HOST CENTRIC MULTIHOMING

In *Host Centric Multihoming*, hosts choose the source and destination addresses of each packet with the aim of doing the best usage of available network resources. As it will be described later, address selection plays a key role in IPv6 multihoming due to the hierarchical nature of PA addressing and the presence of *ingress filters*.

3.1 Destination Address Selection

Before establishing a new communication, a host has to obtain the destination address to be used. Commonly, destination addresses are obtained from the DNS. DNS answers can contain multiple addresses for the same destination. But the DNS is not able to indicate which of these addresses is preferred to be used.

RFC 3484 [Draves, 2003] describes a destination address selection algorithm. While not very smart by default, this algorithm provides, at least, a way to avoid choosing the destination address randomly from the set obtained from the DNS.

To work smartly, this algorithm must be fed with policies. Policies can stem from the administrative policy of the site, and eventually be based on network state information.

Destination address selection influences the path packets will take in their way to the destination.

3.2 Source Address Selection

As multihomed hosts have several IPv6 addresses assigned, once the destination address is chosen, a source address selection algorithm must be performed.

The selected source address will be used as destination address for the reply traffic coming from the remote host. Thus, due to IPv6's hierarchical addressing architecture, the source address used in outgoing traffic determines the path the incoming traffic will follow.

As previously stated, ingress filters check that each packet's source address belongs to the prefix delegated to the site by the ISP. Packets not passing the check are dropped by the ISP's access router.

In order to avoid packets from being dropped by ingress filters, the routing system has to conduct the traffic to the proper *site-exit router*. This implies that the source address selection also determines the path of the outgoing traffic.

Given these considerations, host-centric multihoming has to deal with:

- Source address selection
- Ingress filtering compatibility mechanisms

There are many proposals for doing source address selection. RFC 3484 establishes a basic algorithm that defaults to the longest prefix match of the candidate source addresses and the destination address. To perform better selections, the algorithm introduces a policy table. However, as in the destination address selection case, this table has to be fed in order to be useful.

There are several ways to fill policy tables up; one of them[0] is a way to do it manually. However, to take full advantage of the policy table, entries should be added and deleted dynamically in reaction to network topology changes. A method for policy distribution from the ISP to the site-exit routers [Matsumoto et al., 2004] using an extension to DHCP for Policy Distribution [Troan and Droms, 2003] is proposed. From the site-exit routers to the hosts this approach proposes the usage of a new DHCPv6 option or an extension to the Router Advertisement message.

The NAROS [de Launois et al., 2003] approach establishes a protocol to ask a server which of the available source addresses is the best for a given destination. The NAROS server, based on administrative policies and complete routing information (e.g. obtained via BGP) can select the best address.

Another problem to solve regarding address selection is how to choose addresses in the presence of network failures such as access link problems or routing problems inside the ISP's network. Possible solutions to this kind of problems are discussed in *Host-Centric IPv6 multihoming* [Huitema, 2004].

There are also several proposals for ingress filtering compatibility mechanisms, which address different scenarios. The simplest mechanism is to relax source address filters to allow traffic sourced from any of the site's prefixes. Perhaps the best long term solution is to use a routing protocol that supports *Source Address Dependent* (SAD) routing [Bagnulo et al., 2004].

Standard routing protocols base their routing decisions on the inspection of the destination address; on the other hand, SAD routing protocols take into account both the source and destination addresses. In this way, the routing protocol itself can convey the packets to the correct site-exit routers, circumventing the ingress filters.

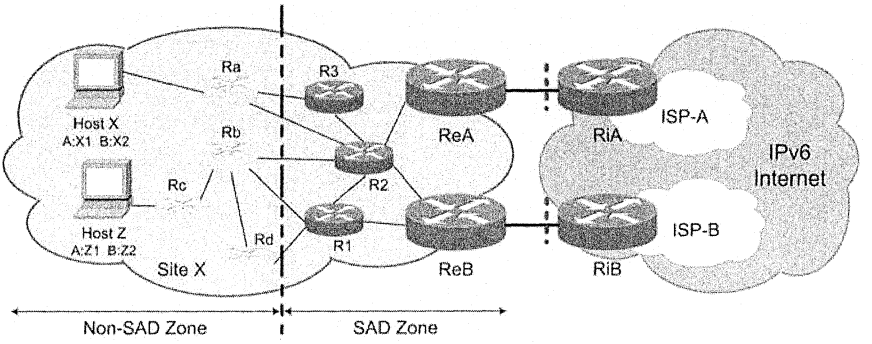

Figure 2. A domain with a mixed SAD and non-SAD routing. Once packets enter the SAD zone, they are conducted towards the proper site-exit router.

In the meantime, other transition mechanisms can be used. A site could have a mix of SAD and non-SAD routing capable routers. The site would be separated into two routing domains, one with SAD routing and the other with non-SAD (i.e. destination based) routing. SAD routing domain has to include at least all the exit routers and must be a connected domain. Non-SAD routing domain does not require routers to be connected, and in fact, it is possible to have several non-SAD routing domains. Once a packet enters the SAD routing domain, it is routed to the correct site-exit router (See Fig. 2).

If this mix is not possible, a full mesh of tunnels between exit routers can be implemented. If a packet arrives to a wrong exit router, it will be tunneled to the correct exit router as depicted in Fig. 3. As usual, a full mesh does not scale well if the number of routers is big and the configuration is done manually.

To handle the burden of tunnel creation there is a proposed solution[0] that detects exit routers and creates tunnels automatically. Each site-exit router is assigned a special anycast address called *site-exit anycast*. These addresses are derived from the site prefixes and routes to them are disseminated by the site's interior routing protocol (IGP). Exit routers can this way discover their peers and associated prefixes.

This method, which is the main topic of the present article, will be described in detail in the next section.

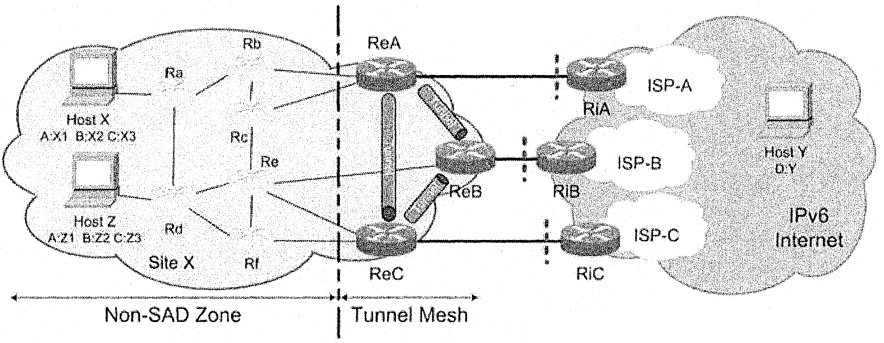

Figure 3. A domain with a full mesh of tunnels. If a packet arrives to an incorrect site-exit router, it is tunneled to the proper site-exit router.

4. AUTOMATIC TUNNELING SOLUTION BASED ON IGP AND ANYCAST ADDRESSES

There are multiple multihoming scenarios depending on the organization's size. We may have big organizations with headquarters around the world, several ISPs servicing in each country and VPNs linking the various country networks. On the other side, we may have a multihomed laptop with wireless LAN and GPRS access.

We focus our work on site multihoming solutions. Thus, we are talking about an organization connected to several ISPs as a whole, and not single laptops connected to casual points.

Big organizations have enough negotiating power to ask the ISPs to relax ingress filters, but medium and small sized organizations don't. The solution discussed in this paper is intended for medium and small sized sites.

The solution solves scenarios as presented in Fig. 4. A site is connected to several ISPs. ISPs run ingress filters on their access routers. In addition, the site's IGP does not perform SAD routing. Therefore, packets won't always arrive to the right exit router for the source address being used.

For instance, if the multihomed *Host X* uses the ISP-C's source address and packets are routed through ISP-A, packets will be dropped by ISP-A's ingress filters.

So a mechanism for ingress filtering compatibility is needed (i.e. a mechanism that reconciles destination based routing with ingress filters). The solution introduced here is based on the automatic tunneling with the *site-exit anycast addresses* mechanism mentioned earlier.

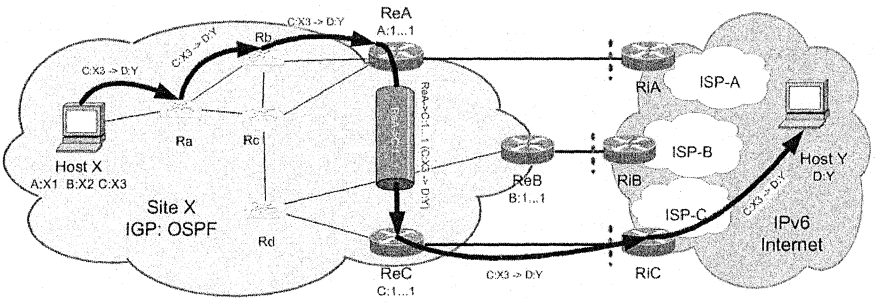

Figure 4. Scenario to be solved by the implementation. Site X is multihomed to three ISPs. If packets from Host X to Host Y reach ReA then they must be tunneled to ReC .

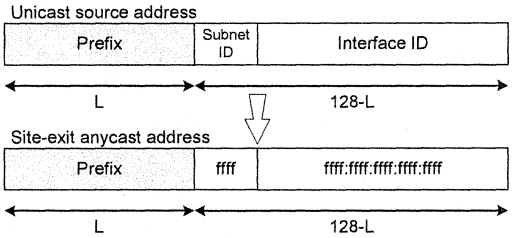

Figure 5. Site-exit anycast addresses can be obtained given a unicast address.

A *site-exit anycast address* is built by appending an "all 1" suffix to a prefix allocated to the site. If a prefix has a length of L, the last 128-L bits of the prefix will be set to 1 (see Fig. 5).

In this proposal, site-exit routers are assigned site-exit anycast addresses for the site prefixes for which they act as exit. Then, host routes generated from anycast addresses are injected into the IGP and disseminated throughout the site. So every router in the domain will know about the existence of these anycast host routes.

In the example shown in Fig. 4, exit router ReC is assigned the site-exit anycast address corresponding to prefix C (denoted as C:1...1 for simplicity). Supposing Site X is running the Open Shortest Path First (OSPF) IGP, a host route for C:1...1 will be injected into OSPF. The same happens with routers ReA and ReB, which are assigned A:1....1 and B:1...1, respectively.

Then, site-exit routers can detect which are the site prefixes tracing site-exit anycast host routes in their Forwarding Information Bases (FIBs). Having this data on hand, it's really simple to obtain the prefix associated to

a site-exit anycast address: just replace the last 128-L bits from 1 to 0 and use L as the prefix length.

With this information, site-exit routers can divert packets toward the correct site-exit router. This diversion is achieved by tunneling packets to the site-exit anycast address associated to the source address prefix, as shown in Fig. 4.

Continuing the example depicted in Fig. 4, host route C:1...1 will be present in ReA's and ReB's FIBs, because it was distributed by OSPF. Then, ReA and ReB will be able to tunnel packets sourced with prefix C towards C:1...1, which is an address assigned to ReB. Both ReA and ReB have no idea where ReC is located, they have just discovered it by means of its site-exit anycast address.

Therefore, there are three tasks to perform at site-exit routers:

- Site-exit router advertisement
- Site-exit router and prefix detection
- Tunnel creation and destruction

It is important to take into account that for this mechanism to work, all the site's prefixes must have the same length (L). This is not an important limitation because it is stated that sites will be assigned 48-bit length prefixes. Therefore, the usual value of L will be 48.

In the example presented in this section, OSPF has been used as IGP. However, the solution is independent of the used IGP.

5. IMPLEMENTATION DETAILS

We have implemented this solution on *Linux* and then ported it to *FreeBSD*. On *Linux*, an *USAGI* project kernel was used because it is the only package that provides IPv6-in-IPv6 tunnels and source based routing. On FreeBSD IPv6-in-IPv6 tunnels are available through the standard ifconfig tool, and source routing was achieved with either IP Filter or OpenBSD Packet Filter tools.

Most of the work was done on *Quagga,* which is a routing software package that implements OSPFv2, OSPFv3, RIP, and BGP routing protocols. It has been ported to several Unix-like operating systems such as GNU/Linux, FreeBSD, NetBSD and Solaris.

Quagga has a modular architecture composed of a main module called *zebra* and a set of secondary modules that implement individual routing protocols. In fact, these modules are called *bgpd, ripd, ospfd, ospf6d, and ripngd.* The main module is the only one that has a strong dependence on the platform, as it is in charge of communication with the kernel. An internal

protocol known as the *Zebra Protocol* is used to communicate protocol modules with *zebra*.

To perform the tests, we relied on the VNUML virtualization tool [VNUML], which allows the creation of complex Linux based virtual network scenarios.

Many aspects influenced the way this implementation was done; among them the following are the most relevant:

- **Multiplatform:** Because *Quagga* is multiplatform, the solution could be easily ported to other OSs.
- **User-Level Implementation:** Since it is not necessary to write kernel code, the development and debugging are simplified. This also makes a more portable implementation.
- **Quagga's Platform:** *Quagga* is not only a routing package, but also a routing platform. It has an extensive function library that developers can take advantage of. *Quagga* is also able to detect routing events such as the addition of routes and interfaces, triggering handlers for these events.
- **Routing Protocol Independence:** Despite the fact that the current implementation is developed and tested using OSPF, developing the solution inside *Quagga's zebra* main module allows the use of other IGPs.
- **Management and Monitoring Interface:** *Quagga* has its own configuration and monitoring interface, which is similar to the command-line interface of *Cisco IOS™*. Having this command-line framework makes adding new commands more simple in order to integrate multihoming configuration in a single user interface.

5.1 Advertisement, Detection and Tunnel Handling

Site-exit routers announce themselves as such by adding the proper *site-exit anycast addresses* to their internal interfaces. These addresses reflect the prefixes for which each router will act as an exit.

To make such announcement, an exit router needs to know the following parameters:

- Length of the site's prefixes (L: usually 48).
- Set of internal interfaces.
- Set of prefixes for which the router will act as an exit.
- Given these parameters the router can:
- Identify the interfaces that the site-exit anycast addresses will be assigned.
- Generate the set of site-exit anycast addresses to advertise.
- Generate source-based default routes for the given prefixes.

In order to receive tunneled packets from other exit routers, the router must also enable a way to receive IPv6-in-IPv6 tunneled packets from any internal source.

After that, host routes derived from the site-exit anycast addresses must be injected into the IGP. In our test implementation, this step is performed by redistributing the routes into OSPF.

An exit router can detect the site prefixes announced by its peers. Because site prefixes are disseminated by the IGP, the detection can be done searching for site-exit anycast entries in the router's FIB. Site-exit anycast address entries in the FIB are represented by host routes. Therefore, the detection process has to look for entries whose prefix length is 128 and whose last 128-L bits are set to 1.

The site prefix associated to the site-exit FIB entry is obtained by setting the last 128-L bits of the entry to 0 and changing the prefix length from 128 to L.

When a new route is added or deleted from the FIB, *zebra* triggers event handlers that execute handling code. So that's the perfect place to put the *site-exit anycast address* detection code. Additionally, the detection is independent of the IGP being used.

Once the prefixes are detected, tunnels can be created. In our implementation, for each detected prefix, a tunnel is created. Each tunnel has the *site-exit* anycast address as the remote endpoint. As the local endpoint address, we use the address assigned to a virtual interface that never goes down (sometimes referred to as *loopback interface*). This way, the tunnel source address does not change over time.

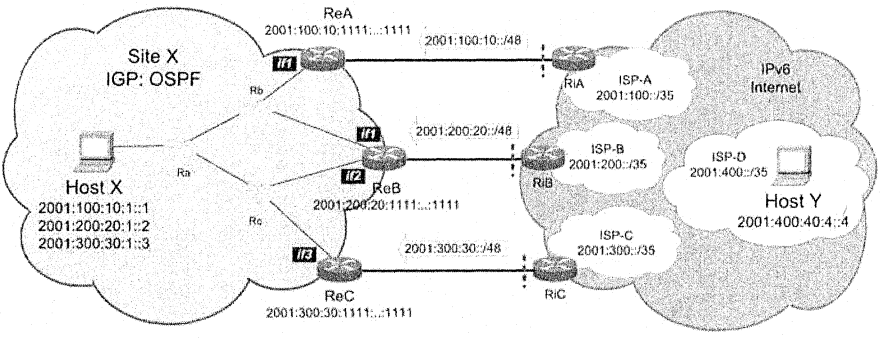

Figure 6. Scenario of the example of automatic tunneling configuration.

But creating the tunnel is not enough for packets to be rerouted to the proper site-exit router. A source-based default route has to be added to the FIB too. This route directs packets with source addresses in the range of the

prefix through the recently created tunnel. As a result of that, packets will travel to the correct site-exit router.

When *zebra* is notified of FIB entry deletions, it also checks if the entry being deleted corresponds to a site-exit anycast route. If so, it destroys the tunnel and deletes the source-based default route.

5.2 Configuration Example

This subsection presents an example of multihoming configuration at site-exit routers. As shown in Fig. 6, a site connects to three ISPs through three exit routers. Each ISP assigns a prefix of length 48 to the site, so the site has three prefixes. Consequently, *Host X* inherits three addresses taken from these prefixes.

ReB's Configuration

```
multihoming
 ipv6 site prefix-length 48
 ipv6 automatic-tunneling
 ipv6 site internal-interface if1
 ipv6 site internal-interface if2
 ipv6 exit-prefix 2001:200:20::/48
```

Figure 7. Site-exit router configuration example for router ReB.

In order to configure multihoming capabilities several commands were added to *Quagga's* command-line interface. A new configuration section was introduced and named `multihoming`. This section holds the following commands:

- `ipv6 automatic-tunneling`
  Enables automatic tunnels usage.
- `ipv6 site prefix-length X`
  Defines the site's prefix length.
- `ipv6 exit-prefix X:X::X:X/M`
  Defines the site's prefixes for which the router acts as exit.
- `ipv6 site internal-interface IFNAME`
  Defines which are the site internal router interfaces.

A command called `show multihoming` was also added to display multihoming status. Fig. 7 presents the relevant configuration commands needed at each exit router for the example in Fig. 6.

6. SUMMARY

The IPv6 multihoming problem has been discussed extensively inside the IETF technical community. The Multi6 IETF working group has set up the foundations of a future solution to the multihoming problem. It is now the task of the new Shim6 to design the solution that will be standardized. This new working group must not only design the so called *Shim6* protocol, but also mechanisms for ingress filtering compatibility, among other tasks.

The article has focused on the *host-centric* multihoming approach, which is based on address selection by the host itself. The implications of source and destination address selection were examined. Source address selection plays a key role because it determines the incoming path for reply traffic and in the presence of ingress filters it also determines the way packets take to leave the site.

Because ingress filters pose many threats to multihoming, there are multiple proposed ways to handle the problem. From relaxing the ingress filters to the setup of a static mesh of tunnels between site-exit routers.

We have implemented one of the proposed ingress filtering compatibility mechanisms. This mechanism uses anycast addresses to advertise site exit routers to their peers. In this way, site exit routers can tunnel packets to the correct exit routers if they are not the correct exit routers for some packets.

The implementation was done on Linux and FreeBSD, using the Quagga routing package. In the tests it performed its functions as expected, advertising and detecting the site exit routers, creating and destroying the tunnels, and redirecting the packets towards the correct exit router. This happened despite the different platform details. One advantage of this implementation is that it was done in user space, that way it was easily ported to FreeBSD from Linux. Deployment of this solution only needs changes on exit routers. At the time being it is not deployable on proprietary commercial routers.

ACKNOWLEDGEMENTS

We would like to thank all the partners involved in the Euro6IX project for their support and cooperative attitude.

REFERENCES

Bagnulo, M. et al. (2004). The Case for Source Address Dependent Routing in Multihoming, Lecture Notes in Computer Science, Volume 3266, 2004.

Bu, Tian, Gao, Lixin, and Towsle, Don (2002). On Characterizing BGP Routing Table Growth. Proceedings of IEEE Global Internet Symposium 2002.

Crocker, D. (2003). Multiple Address Service for Transport (MAST): An Extended Proposal. Internet draft draft-crocker-mast-proposal-01. IETF. September 2003.

de Launois, C. et al. (2003) The NAROS Approach for IPv6 Multihoming with Traffic Engineering, Proceedings of QoFIS 2003, Lecture Notes in Computer Science

Draves, R. (2003). Default Address Selection for Internet Protocol version 6 (IPv6), RFC 3484, IETF, 2003, http://www.ietf.org/rfc/rfc3484.txt

Dunmore, M. (Editor) (2003). Report on IETF Multihoming Solutions, version 2. Deliverable 4.5.1, 6NET Project. http://www.6net.org/publications/deliverables/D4.5.1v2.pdf

Hagino, J. and Snyder, H. (2001). IPv6 Multihoming Support at Site Exit Routers. RFC 3178, IETF, http://www.ietf.org/rfc/rfc3178.txt

Huitema, C. (2004). Host-Centric IPv6 Multihoming. draft-huitema-multi6-hosts-03, IETF, Work in progress

Matsumoto, A. et al. (2004) Source Address Selection Policy Distribution for Multihoming, draft-arifumi-multi6-sas-policy-dist-00, IETF Work in progress.

Moskowitz, R., Nikander, P., Jokela, P. and Henderson, T. (2005). Host Identity Protocol. Internet draft draft-ietf-hip-base-02. IETF. February 2005.

multi6. Site Multihoming in IPv6 (multi6). IETF Working Group. http://www.ietf.org/html.charters/multi6-charter.html

Tapiador, A. et al. (2004). A simple host centric solution for a network research multihoming environment. In Proceedings of EUNICE 2004

Teraoka, F., Ishiyama, M. and Kunishi, M. (2003). LIN6: A Solution to Multihoming and Mobility in IPv6. Internet draft draft-teraoka-multi6-lin6-00. IETF. December 2003.

Troan, O. and Droms, R. (2003), IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version, RFC 3633, IETF. http://www.ietf.org/rfc/rfc3633.txt

VNUML Tool. http://www.dit.upm.es/vnuml