Multi-Homing Tunnel Broker

Marcelo Bagnulo, Juan Fco. Rodríguez-Hervella, Alberto García-Martínez, Arturo Azcorra
Universidad Carlos III de Madrid. Av. Universidad, 30 – Leganés, 28911 Madrid, SPAIN
{marcelo,jrh,alberto,azcorra}@it.uc3m.es

Abstract

A proper support for communications has to provide fault tolerance capabilities such as the preservation of established connections in case of failures. Multi-homing addresses this issue, but the currently available solution based in massive BGP route injection presents serious scalability limitations, since it contributes to the exponential growth of the BGP table size. An alternative solution based on the configuration of tunnels between the multihomed site exit routers and the ISP border routers has been proposed for IPv6 in RFC 3178. However, the amount of manual configuration imposed by this solution on the ISP side prevents its wide adoption. In particular, this solution requires at the ISP the manual configuration of a tunnel endpoint per each multihomed client that it serves. In this paper we present a Multi-Homing Tunnel Broker (MHTB) that provides automatic creation of the tunnel endpoint at the ISP side.

1. Introduction[1]

Since global connectivity has become a critical resource for organizations, sites attempt to improve the qualities of their communication facilities obtaining Internet connectivity through two or more providers. However actual benefit obtained does not only depend on the number of different physical links hired, but it also depends on the way end-hosts and the routing system cope with this multiple path information. In the IPv4 Internet, several multi-homing solutions have been deployed, providing different levels of benefit. The multihoming solution most widely adopted in IPv4 Internet is based on the route injection of multihomed site prefixes into the global routing system. While this solution provides good fault tolerance capabilities, its direct adoption in IPv6 would hinder IPv6 routing system scalability, since it is not compatible with hierarchical route aggregation. An alternative solution that presents better scalability features has been presented in RFC 3178 [1]. RFC3178-style multihoming is compatible with provider aggregation of addresses and routes and it also preserves routing system scalability since it does not introduce additional routes in the global routing table. RFC 3178 multihoming is based on the configuration of tunnels between the end-site exit routers and the ISP border routers. These tunnels carry the traffic addressed to the site when the direct link between one of the ISPs and the site is down. However, there are some concerns regarding the deployability of this solution, since its adoption requires the manual configuration by the ISP of a tunnel endpoint per each one of its multihomed customers. Considering that this solution is aimed to provide a widely adopted solution, the manual configuration of the tunnels may impose a high management workload at the ISPs.

This paper proposes the deployment of Multi-Homing Tunnel Brokers to configure the ISP side of the tunnel automatically. Such mechanism would diminish the manual configuration required at the ISP side, rendering the RFC 3178 style multihoming much more appealing to the ISPs. We will first describe the currently deployed IPv4 multi-homing solution and its limitations. Then we will analyse the multi-homing solution presented in RFC 3178. Next we will present the tunnel broker model and how it simplifies the management of the tunnels required by RFC 3178-style multihoming. Related work is discussed in the following section. Finally, we will present the implementation of the solution that we have deployed and some final remarks.

2. Background

A site is *multi-homed* when it obtains Internet connectivity through two or more service providers. Through multi-homing an end-site improves the fault tolerance of its connection to the global network and it can also optimise the path used to reach the different networks connected to the Internet.

[1] This work was supported by the SAM (Advanced Servers with Mobility) project, funded by the Spanish National Research and Development Programme as TIC2002-04531-C04-03.

2.1 IPv4 multi-homing

In IPv4, the most widely deployed multi-homing solution is based on the announcement of the site prefix through all its providers, as it is described in figure 1.

In this solution, the site *S* obtains a prefix allocation either directly from the corresponding RIR (Regional Internet Registry) or from one of its providers (*ISPA* or *ISPB*). Then, the site announces this prefix (*PSite*) to its providers using BGP. *ISPA* and *ISPB* announce the prefix to their providers and so on, so that the route is announced in the Default Free Zone. It must be noted that even when *PSite* is part of a provider aggregate, PSite has to be announced separately in order to avoid that the longest prefix match rule diverts all the traffic to the provider announcing the more specific route.

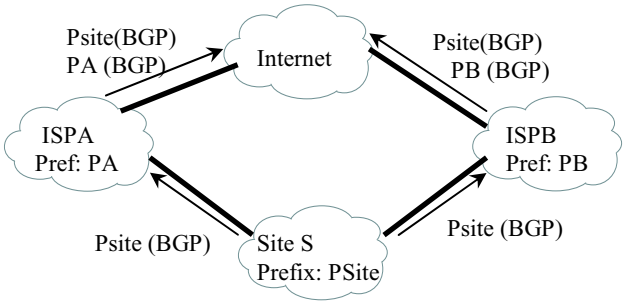

Figure 1. IPv4 multi-homing solution

This mechanism presents many useful properties, such as good fault tolerance capabilities, including preserving established connections throughout an outage, since alternative routes are used without end host perception. However, each multi-homed site using this solution contributes with an additional route to the Default Free Zone routing table, imposing extra stress to already oversized routing tables. For this reason, more scalable multi-homing solutions are required for IPv6.

2.2. Provider aggregation and multi-homing

In order to reduce routing table size, the usage of some form of provider aggregatable addressing is needed. This means that sites obtain prefixes that are part of their provider allocation, so that the providers only announce the aggregated address block to their non-client peers, as it is illustrated in figure 2. Most aggressive aggregation than the one currently obtained for IPv4 is achieved by aggregating end-sites into their provider prefixes, so provider aggregation of end-sites is deemed necessary. Further aggregation, i.e. direct provider prefixes aggregated into transit provider prefix, provides much less aggregation benefits while it does present some deployment challenges such as in multi-homed provider scenarios, so its adoption remains uncertain.

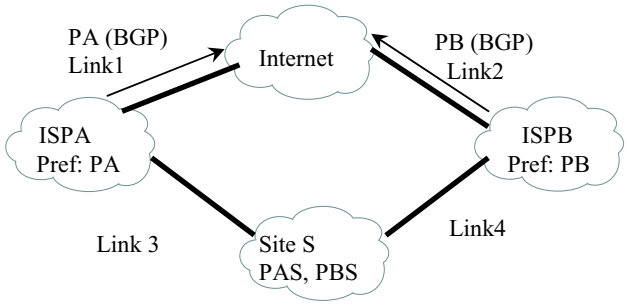

Figure 2. End-site prefix aggregation

When provider aggregation of end-site prefixes is used, end-site host interfaces obtain one IP address from each allocation, in order to benefit from multi-homing capabilities, since ISPs will only forward traffic addressed to their own aggregate. The large address space available in IPv6 enables such usage. Note that ISPs only announce their aggregated address block to their providers, and they do not announce prefixes belonging to other ISPs.

This configuration presents several concerns as we will present next.

Increased connection establishment time in case of failure. When *Link 1* or *Link 3* becomes unavailable, addresses containing the *PAS* prefix are unreachable from the Internet. New incoming (from the site perspective) connections using *PAS* addresses will suffer from an increased establishment time, due to the discovery of unreachable addresses by the originator node.

Established connections will not be preserved in case of outage. If *Link 1* or *Link 3* fails, already established connections that use addresses containing the *PAS* prefix will fail, since packets addressed to the *PAS* aggregate will be dropped because there is no route available for this destination.

Ingress filtering incompatibility. Ingress filtering [2] is a widely used technique for preventing the usage of spoofed addresses. However, in the described configuration, its usage presents additional difficulties for the source address selection mechanism and the intra-site routing systems, since the exit path and the source address of the packet must be coherent with the incoming path, in order to bypass ingress filtering mechanisms.

The presented difficulties show that additional mechanisms are needed in order to build a multihoming solution compatible with the usage of provider aggregatable addresses, while still providing equivalent benefits to those of the IPv4-style multi-homing solution.

2.3. Multi-homing requirements

In order to design IPv6 multihoming solutions, the set of requirements for the solution have been defined by the `multi6` working group at the IETF. The initial requirements identified for multi-homing are [3]:

Redundancy and reliability: A multi-homing architecture is built to improve Internet connection reliability, so fault tolerance is a key issue in any proposed mechanism. A multihoming solution has to provide protection against the following failure modes: physical and logical link breakdowns, routing protocol malfunction, and ISP or exchange crash.

Transport session survivability: In multihomed environments, transport connections have to be preserved when an outage occurs along the path that is being used for routing the packets.

Load sharing: Distribution of load among available links is another key issue in a multi-homed environment.

Policing: The multihoming solution has to provide the required tools to implement policing, so that traffic is routed according to internally defined policies. For instance, for cost optimization, depending on the SLA agreed, cost and quality of different connections may vary among ISPs. Cost-aware selection mechanisms are needed to fulfil the requirements of multi-homed organizations.

Simplicity is a relevant condition if the architecture is meant to be deployed. The fewer the changes in existing protocols and mechanisms that are introduced, the faster the solution would be developed.

Scalability has been a major concern in IPv6 definition so proposed multi-homing architectures must adhere to this design criterion. In particular, routing scalability provided by ISP-based aggregation must be preserved. Other scalability issues such as the manageability of the solution should also be considered.

3. Overview of the RFC 3178 multi-homing solution

RFC 3178 [1] describes a solution to provide site multi-homing support in IPv6. This solution uses tunnels between the different ISPs and the multi-homed site to provide alternative paths in case that one of the exit links is down, protecting the multi-homed site from outages in the direct link with its providers.

The mechanism for multi-homing support described in RFC3178 is illustrated in the next figure:

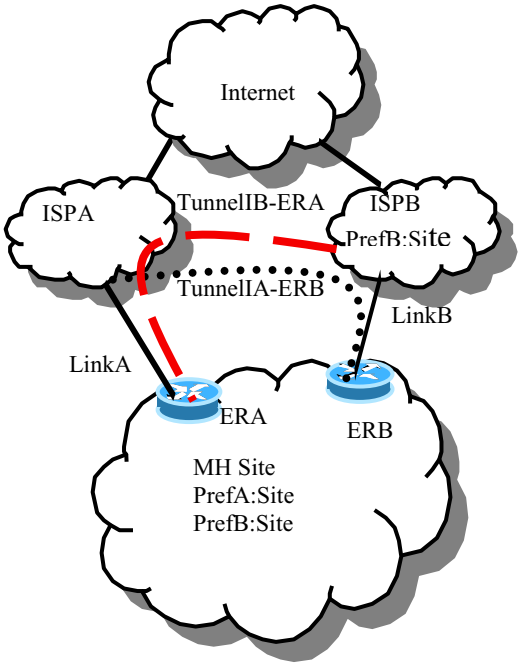

Figure 3. RFC 3178 multihoming

The multi-homed site has two providers, *ISPA* and *ISPB*, that have delegated *PrefA:Site::/n* and *PrefB:Site::/m* respectively. In the depicted scenario, the multi-homed site has only two providers, but the solution is valid to more general scenarios that include more than two providers. It is assumed that hosts within the multi-homed site configure at least one address per provider's prefix obtained.

In order to obtain fault tolerance capabilities, RFC 3178 proposes the creation of two tunnels:

TunnelIA-ERB: from the *ISPA*'s border router to the site exit router *ERB*

TunnelIB-ERA: from the *ISPB*'s border router to the site exit router *ERA*

The resulting behaviour is that when one of the two exit links fails, packets are routed through the correspondent tunnel. That is, if *linkA(linkB)* fails, packets arriving to *ISPA*(*ISPB*) addressed to *PrefA:Site::/n(PrefB:Site::/m)* are routed through *TunnelIA-ERB* (*TunnelIB-ERA*) to the multi-homed end site.

This configuration provides fault tolerance capabilities, including the preservation of established communications, when one of the site exit links fails.

However, the wide adoption of RFC 3178 multi-homing solution implies the manual configuration of numerous tunnels on the ISPs, which may impose an important workload in ISP network administrators. We will next present the tunnel broker framework and then we will propose the usage of a Multi-Homing Tunnel Broker

(MHTB) to automatically configure the ISP tunnel endpoint in order to ease the adoption of the solution.

4. The Tunnel Broker model

RFC 3053 [4] presents a general tunnel broker model and its particular application to the creation of IPv6 over IPv4 tunnels for the Internet's transition to IPv6. Since the tunnels used in RFC 3178 are IPv6 in IPv6 tunnels used for redundancy, the particular implementation details will differ in the two configurations. However, the general model of the tunnel broker presented in RFC 3053 can still be applied to the multi-homing environment.

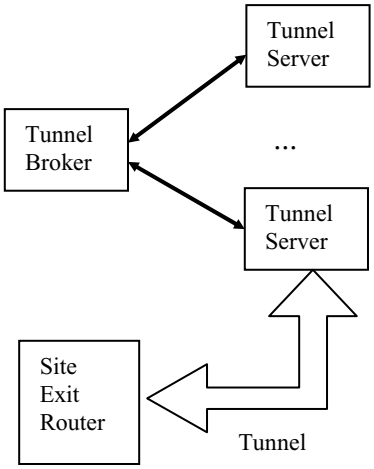

Figure 4. Tunnel Broker framework

The tunnel broker model presented in RFC 3053 is illustrated in the next figure and it consists of the following components:

The *Tunnel Broker* (*TB*) is the element to which the end-users connect themselves to create, modify and delete tunnels. Then the TB communicates with one or several Tunnel Servers to actually create the tunnels requested by the users.

The *Tunnel Server* (*TS*) is the server's tunnel endpoint that is created, modified or deleted upon reception of a request from the TB.

5. The multi-homing Tunnel Broker service

In order to provide a Tunnel Broker Service for the RFC3178 multi-homing solution, a Tunnel Broker and one or more Tunnel Servers are required.

5.1. The Tunnel Broker service

The TB will receive user requests to create tunnels. Potential users of this service are all the clients of the ISP. We assume that the ISP clients have a commercial relationship with the ISP, so that the ISP can identify its clients and the prefix that the ISP has assigned to them. We also assume that, because of the existent business relationship between the ISP and its customer, the ISP has created the appropriate means to identify its clients through the network, such as a user name and a password or a public key certificate.

So, the ISP customer will send a tunnel creation request to the TB. The TB can accept requests through different type of interfaces, for instance the TB can accept request submitted through HTTP. Clients submitting requests have to properly identify themselves through existent means.

The requests have to contain at least the following information:

- Client identification.
- The IPv6 address of the client side endpoint of the tunnel.
- The IPv6 prefix for which a backup route through the tunnel will be created.
- Request authorization information generated through available means, such as client's password or client's private/public key plus certificate.

Upon the reception of a request the TB will:

- Verify client identity.
- Verify authorization information.
- Verify that the prefix included in the request is contained in the address range that the ISP has delegated to this particular client.
- Send a configuration order to the appropriate TS to configure the requested tunnel.
- Inform the client about the endpoint of ISP side of the tunnel. Additionally, detailed configuration information for popular routers can also be provided to the client.

5.2. The Tunnel Server

The Tunnel Server service can be placed in different devices within the ISP network.

An option is to use the exit router connecting the ISP to the client as the tunnel server. In this case, packets will always be forwarded to this exit router, and if the route through the direct link is not available, the exit router will forward the packets through the tunnel interface. This option requires the TB to be capable of communicating with the exit routers through the protocol selected (RSH, SNMP, others).

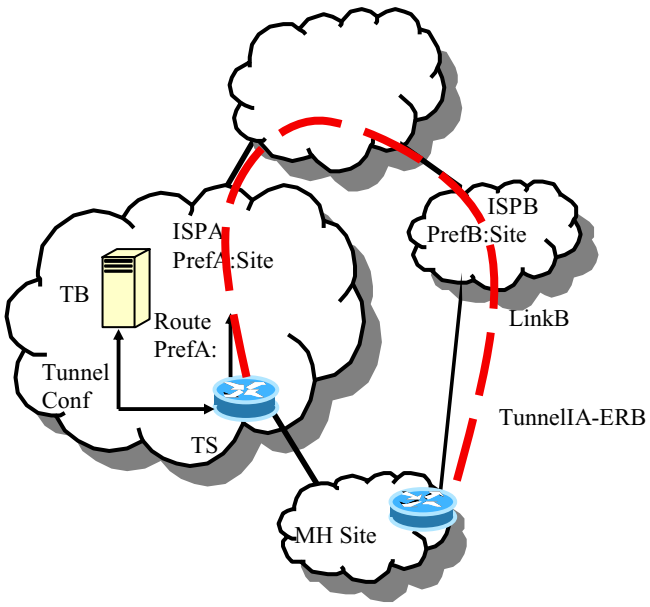

Figure 5: TS collocated with exit router

Another option is to place specific TS devices, separated from the existent routing infrastructure. In this case, the TB will configure the tunnel in this specific TS device. The TS will then, upon the reception of this configuration orders, create the tunnel and will also start announcing the client prefix through a route with a preference lower than the one contained in the route announced by the exit router into the provider network.

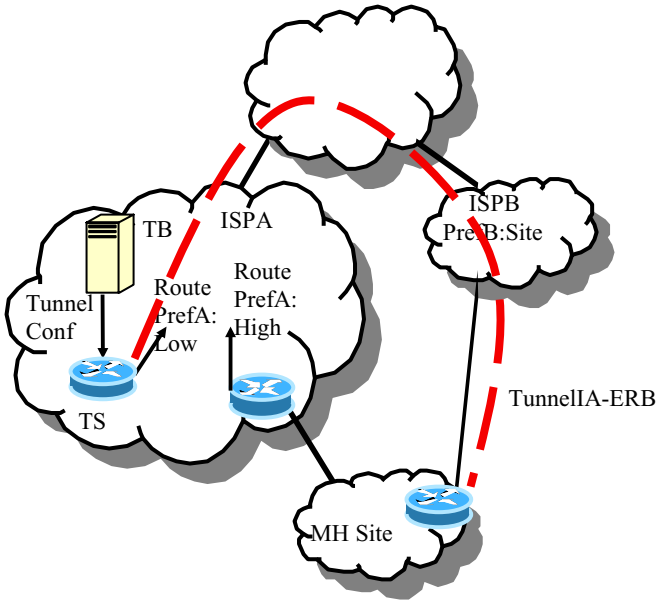

Figure 6. Specific TS devices

5.3. Extending the protection domain

A major limitation of the RFC 3178 multihoming solution is its limited fault tolerance, since it only protects from outages occurred in the direct link between the multihomed site and one of its direct providers (*LinkA* or *LinkB* in the figure). Other failure modes still affect the site. For instance a failure in the link between the ISPs and its upstream ISP (*LinkC* in the next figure) would affect the multihomed site if communications were being routed through this ISP.

A possibility to extend the protection domain would be to create tunnels between the respective upstream ISPs and the multihomed site, as described in the next figure.

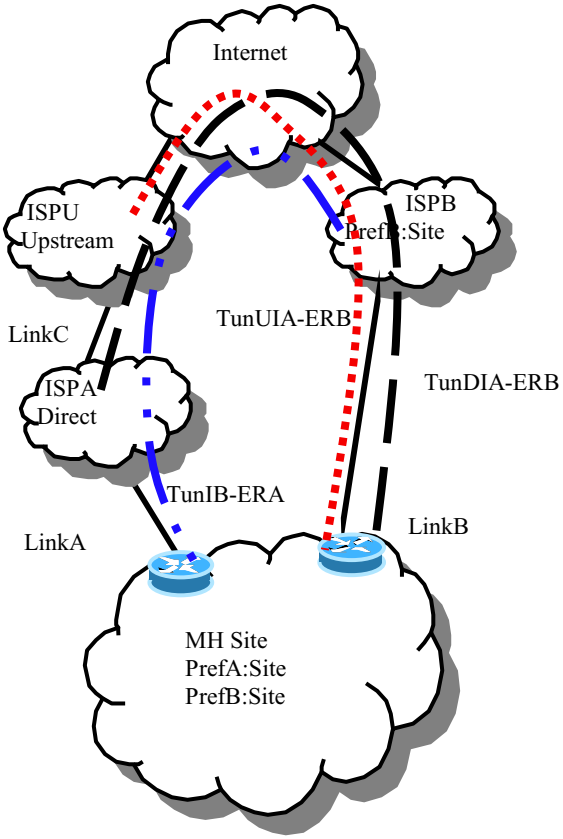

Figure 7. Extended protection

In this case additional tunnels would be created as *TunUIA-ERB* in the figure, between the upstream ISP and the multihomed site.

A minor issue has to be considered since the route through the tunnel would be a route to the site prefix which will be more specific than the direct route to the aggregated block of the ISP. The resulting behaviour would be that all the traffic is normally routed through the tunnel instead of using the direct route, which is not the desired behaviour. The solution for this is to deaggregate

the ISP block in the internal routing, so that more specific routes to those multihomed sites are announced.

The other problem that has to be considered is the amount of tunnels that have to be configured in the upstream ISP. Note that the upstream ISP (*ISPU*) will have to configure one tunnel per each multihomed customer of its customer ISPs. So, in this scenario, the utilization of the Multi-Homing Tunnel Broker would greatly simplify the creation and management of the required tunnels, fostering the adoption of the solution. Moreover, the Multi-Homing Tunnel Broker of the direct ISP (*ISPA*) can directly communicate with the Multi-Homing Tunnel Broker of the upstream ISP (*ISPU*) when the first is contacted by the multihomed customer. The process would be the following:

First, the multihomed customer contacts the Multi-Homing Tunnel Broker of its direct ISP (*ISPA*). At this point, the TB contacts the TS and configures the requested tunnel. Additionally, the TB of the direct ISP (*ISPA*) automatically contacts the upstream ISP (*ISPU*) Multi-Homing Tunnel Broker and it requests the configuration of a tunnel in behalf of the multihomed site. The upstream Multi-Homing Tunnel Broker validates the request as usual and if accepted, it configures the requested tunnel. It then returns information about the IP address of the upstream ISP tunnel endpoint to the direct ISP Multi-Homing Tunnel Broker. Then, the direct ISP Multi-Homing Tunnel Broker informs the client about both tunnel endpoints, the one created by the direct ISP (*ISPA*) and the one created by the upstream ISP (*ISPU*). The multihomed site then configures its side for both tunnels.

This mechanism allows extending the protection domain of the solution to upstream ISPs. It should be noted that the usage of Multi-Homing Tunnel Broker makes all the process transparent to ISP managers, since the tunnel endpoint creation is automatic.

6. Implementation

The tunnel broker implementation, composed of a TB module and TS scripts, is based on the Linux open source implementation developed by Groth [5] for establishing IPv6 over IPv4 tunnels. It makes use of Apache 1.3 with IPv6 support, PHP-4.3.4 and MySQL-4.0.18 open source components.

The TB implementation consists of a set of PHP pages, which generate the interface that the customers can access via HTTPS, and a database system to store both user information and configuration data. These elements provide the basic TB functionality. The TS functionality for establishing the tunnels is provided by a set of scripts which are periodically run in the background, scripts that can be executed on a different system. The main changes to support the specification of the Multi-homing Tunnel Broker are the IPv6-in-IPv6 tunnel management, and the modification of the web front-end. The adapted version works at UNIX systems, and generates configurations tailored for both Linux and BSD systems.

The administrator of the client site will use a generic web browser to access the TB. The process of requesting a tunnel proceeds as follows: the administrator connects with the TB and provides its login and password. Once the client administrator has successfully signed up, it has to provide the IPv6 address which stands for the client tunnel endpoint. The TB will provide the user with the needed information to set up the tunnel in the customer premises. The background TS scripts will access to the request information that has been stored in the database and will configure the provider's router taking into account the information delivered by the customer.

A periodic script which 'pings' the remote endpoint is used to detect errors in the tunnel configuration process.

The three Tunnel Broker components are described below.

6.1. Database

The database system contains several tables. The "user" table holds the user information required to access to the service, including the user's authorization data. This information has been previously generated as a result of an agreement between the user and the provider. The "allocation" table contains one entry per user storing information about the prefixes that have been delegated to the customer, the maximum number of tunnels that are allowed for the customer, and the number of tunnels that has been allocated for him. The restriction in the number of tunnels facilitates the control of the resource usage at the provider. The "allocation" table also stores the address that will be used for the tunnel end-point at the provider.

The "tunnel" table holds information about each configured tunnel, containing the prefix that the client has requested for being rerouted through the tunnel in case of link failure (that can be more specific than the one stored in the "allocation" table), and the address of the end-point at the client. Finally, the "statistics" table keeps information about the usage of all the tunnel interfaces. This table is updated by a periodic script.

6.2. Web front-end

The web front-end accesses to the database to authenticate the user, checks if the address range for which the alternate path has been requested is correct, and checks the number of tunnels that have been provided. If all the conditions hold, the front-end updates the database with the client's tunnel end-point address, and accesses to the allocation table to generate a web page that contains

the provider's tunnel end-point address. This information could be completed with the detailed configuration that the client should include for different popular router interfaces.

Figure 8. Tunnel Broker web front-end

6.3 TS scripts

The implementation makes use of a set of scripts to provide the TS functionality. This permits the distribution of tasks between several machines, and in particular, several routers can serve as tunnel end-points, since the database allows the specification of different local tunnels for different users. The scripts that have been implemented are:

- A script to create tunnels.
- A script to gather statistics, which can be displayed in a web page.
- A script to check tunnel end-point reachability.

7. Related work

Multiple solutions have been presented during the last few years to provide multihoming support.

Some of these solutions are based on the creation of a new name space to provide a location-independent identity for the hosts. This would allow using Provider Assigned IP addresses as a mean to locate a node, and to use the new namespace to identify the node independently of the path that is being used to reach it. With such separation, transport and application layer would identify the other endpoint of the communication through its location-independent identifier, which would remain constant even if the path used to reach the multihomed host changes because of outages. Some of these solutions propose the creation of a cryptographically based name space in order to provide the required security features. Examples of such solutions are HIP [6] and SIM [7] among others. Other solutions, like NOID [8] or ODT [9], just use a different IP address as identifier, which will remain constant even if the path changes.

While these solutions may provide better fault tolerance capabilities than RFC 3178, their adoption requires the modification of both ends of a communication. This means that not only the nodes within the multihomed site have to adopt the mechanism, but also the nodes outside the multihomed site. It should be noted that there are not direct benefits for this nodes to adopt the solution, since they are not the ones who are multihomed, so the adoption of this type of solutions remains uncertain. RFC 3178 only requires the configuration of a tunnel, process that can be greatly simplified by the use of the multihoming tunnel broker proposed in this paper.

Another possible approach is to modify the transport layer, essentially TCP, in order to support multiple IP addresses per connection. The current TCP specification identifies a connection by the combination of source and destination addresses, and source and destination port numbers. The idea is to modify TCP in order to recognize packets with alternative addresses as belonging to the established connection [10]. Note that there already exist transport protocols as DCCP [11] or SCTP [12] that support this type of functionality, so its usage in multihomed environments should be straightforward. However, these solutions would only provide multihoming benefits to those applications that run over the modified transport layer, and not for applications running over "legacy" TCP or UDP. RFC 3178 multihoming provides protection to all applications independently of the transport protocol that is being used to carry the traffic.

8. Conclusions

In this paper we have presented a mechanism to simplify the management of the tunnels required to deploy the multihoming solution described in RFC 3178. This solution provides multihoming support by configuring tunnels between the multihomed site and its direct providers. Such tunnels are used to provide fault tolerance when the direct link between the multihomed site and the ISP fails. While this multihoming solution is straightforward, it implies a considerable amount of manual configuration of tunnel endpoints, especially relevant at the ISP site. The Multi-Homing Tunnel Broker proposed in this paper provides a mechanism to perform tunnel configuration in an automatic fashion, facilitating RFC 3178 multihoming solution adoption. An

implementation of the Multi-Homing Tunnel Broker has been developed as a proof-of-concept. It is expected to be a framework to perform more research over RFC 3178. Further work includes separating the TB and the TS, as well as developing a communication system between the TB and either the inter-domain or intra-domain routing system. The aim would be to get a fully automated tool to build up the quite complex process defined in RFC 3178.

9. References

[1] Hagino, J. and H. Snyder, "IPv6 Multihoming Support at Site Exit Routers", RFC 3178, October 2001.

[2] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", RFC 2267, January 1998.

[3] Abley, J., Black, B. and V. Gill, "Goals for IPv6 Site-Multihoming Architectures", RFC 3582, August 2003.

[4] Durand , A., Fasano , P., Guardini , I. and D. Lento, "IPv6 Tunnel Broker", RFC 3053, January 2001.

[5] http://sourceforge.net/projects/tunnelbroker, Duane Groth.

[6] P. Nikander, J. Arkko, End-Host Mobility and Multi-Homing with Host Identity Protocol, Internet draft (work in progress), December 2003.

[7] E. Nordmark, Strong Identity Multihoming using 128 bit Identifiers (SIM/CBID128), Internet draft (work in progress), October 2003.

[8] E. Nordmark, Multihoming without IP Identifiers, Internet draft (work in progress), October 2003.

[9] I. van Beijnum, On Demand Tunneling For Multihoming, Internet draft (work in progress), January 2004.

[10] A. Matsumoto, M. Kozuka, K. Fujikawa, Y. Okabe,Y., "TCP Multi-Home Options", Internet draft (work in progress), September 2003.

[11] Kohler, E., M. Handley, S. Floyd, J. Padhye, "Datagram Congestion Control Protocol (DCCP)", Internet Draft, Work in progress, 30 June 2003.

[12] R. Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., Paxson, V., "Stream Control Transmission Protocol", RFC 2960, October 2000.