Network Working Group
Request for Comments: 3053
Category: Informational

A. Durand
SUN Microsystems, Inc
P. Fasano
I. Guardini
CSELT S.p.A.
D. Lento
TIM

January 2001

IPv6 Tunnel Broker

Status of this Memo

Copyright Notice

Abstract

The IPv6 global Internet as of today uses a lot of tunnels over the existing IPv4 infrastructure. Those tunnels are difficult to configure and maintain in a large scale environment. The 6bone has proven that large sites and Internet Service Providers (ISPs) can do it, but this process is too complex for the isolated end user who already has an IPv4 connection and would like to enter the IPv6 world. The motivation for the development of the tunnel broker model is to help early IPv6 adopters to hook up to an existing IPv6 network (e.g., the 6bone) and to get stable, permanent IPv6 addresses and DNS names. The concept of the tunnel broker was first presented at Orlando's IETF in December 1998. Two implementations were demonstrated during the Grenoble IPng & NGtrans interim meeting in February 1999.

1. Introduction

The growth of IPv6 networks started mainly using the transport facilities offered by the current Internet. This led to the development of several techniques to manage IPv6 over IPv4 tunnels. At present most of the 6bone network is built using manually configured tunnels over the Internet. The main drawback of this approach is the overwhelming management load for network administrators, who have to perform extensive manual configuration for each tunnel. Several attempts to reduce this management overhead

have already been proposed and each of them presents interesting advantages but also solves different problems than the Tunnel Broker, or poses drawbacks not present in the Tunnel Broker:

- the use of automatic tunnels with IPv4 compatible addresses [1] is a simple mechanism to establish early IPv6 connectivity among isolated dual-stack hosts and/or routers. The problem with this approach is that it does not solve the address exhaustion problem of IPv4. Also there is a great fear to include the complete IPv4 routing table into the IPv6 world because this would worsen the routing table size problem multiplying it by 5;

- 6over4 [2] is a site local transition mechanism based on the use of IPv4 multicast as a virtual link layer. It does not solve the problem of connecting an isolated user to the global IPv6 Internet;

- 6to4 [3] has been designed to allow isolated IPv6 domains, attached to a wide area network with no native IPv6 support (e.g., the IPv4 Internet), to communicate with other such IPv6 domains with minimal manual configuration. The idea is to embed IPv4 tunnel addresses into the IPv6 prefixes so that any domain border router can automatically discover tunnel endpoints for outbound IPv6 traffic.

The Tunnel Broker idea is an alternative approach based on the provision of dedicated servers, called Tunnel Brokers, to automatically manage tunnel requests coming from the users. This approach is expected to be useful to stimulate the growth of IPv6 interconnected hosts and to allow early IPv6 network providers to provide easy access to their IPv6 networks.

The main difference between the Tunnel Broker and the 6to4 mechanisms is that the they serve a different segment of the IPv6 community:

- the Tunnel Broker fits well for small isolated IPv6 sites, and especially isolated IPv6 hosts on the IPv4 Internet, that want to easily connect to an existing IPv6 network;

- the 6to4 approach has been designed to allow isolated IPv6 sites to easily connect together without having to wait for their IPv4 ISPs to deliver native IPv6 services. This is very well suited for extranet and virtual private networks. Using 6to4 relays, 6to4 sites can also reach sites on the IPv6 Internet.

In addition, the Tunnel Broker approach allows IPv6 ISPs to easily perform access control on the users enforcing their own policies on network resources utilization.

This document is intended to present a framework describing the guidelines for the provision of a Tunnel Broker service within the Internet. It does not specify any protocol but details the general architecture of the proposed approach. It also outlines a set of viable alternatives for implementing it. Section 2 provides an overall description of the Tunnel Broker model; Section 3 reports known limitations to the model; Section 4 briefly outlines other possible applications of the Tunnel Broker approach; Section 5 addresses security issues.

2. Tunnel Broker Model

Tunnel brokers can be seen as virtual IPv6 ISPs, providing IPv6 connectivity to users already connected to the IPv4 Internet. In the emerging IPv6 Internet it is expected that many tunnel brokers will be available so that the user will just have to pick one. The list of the tunnel brokers should be referenced on a "well known" web page (e.g. on http://www.ipv6.org) to allow users to choose the "closest" one, the "cheapest" one, or any other one.

The tunnel broker model is based on the set of functional elements depicted in figure 1.

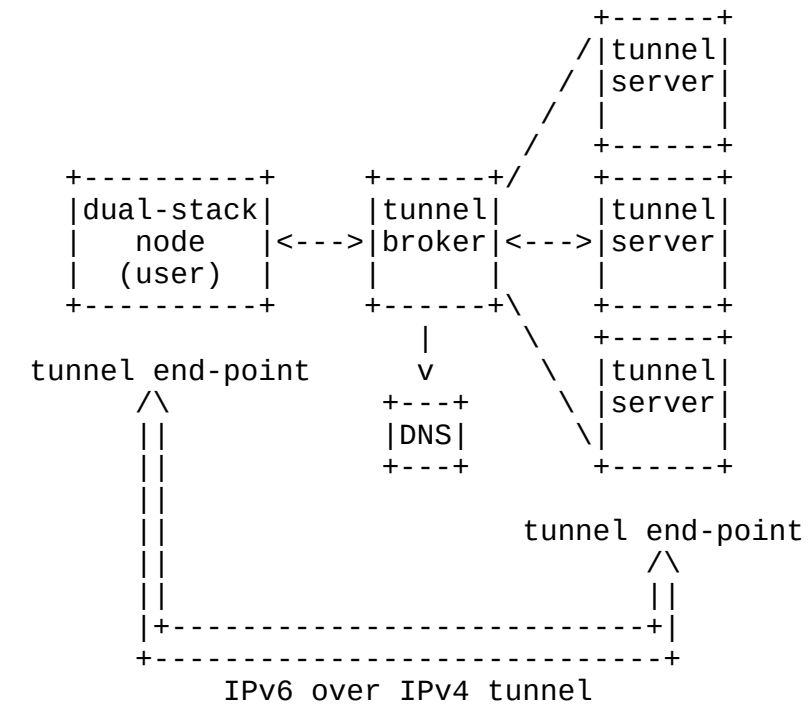

Figure 1: the Tunnel Broker model

2.1 Tunnel Broker (TB)

The TB is the place where the user connects to register and activate tunnels. The TB manages tunnel creation, modification and deletion on behalf of the user.

For scalability reasons the tunnel broker can share the load of network side tunnel end-points among several tunnel servers. It sends configuration orders to the relevant tunnel server whenever a tunnel has to be created, modified or deleted. The TB may also register the user IPv6 address and name in the DNS.

A TB must be IPv4 addressable. It may also be IPv6 addressable, but this is not mandatory. Communications between the broker and the servers can take place either with IPv4 or IPv6.

2.2 Tunnel server (TS)

A TS is a dual-stack (IPv4 & IPv6) router connected to the global Internet. Upon receipt of a configuration order coming from the TB, it creates, modifies or deletes the server side of each tunnel. It may also maintain usage statistics for every active tunnel.

2.3 Using the Tunnel Broker

The client of the Tunnel Broker service is a dual-stack IPv6 node (host or router) connected to the IPv4 Internet. Approaching the TB, the client should be asked first of all to provide its identity and credentials so that proper user authentication, authorization and (optionally) accounting can be carried out (e.g., relying on existing AAA facilities such as RADIUS). This means that the client and the TB have to share a pre-configured or automatically established security association to be used to prevent unauthorized use of the service. With this respect the TB can be seen as an access-control server for IPv4 interconnected IPv6 users.

Once the client has been authorized to access the service, it should provide at least the following information:

- the IPv4 address of the client side of the tunnel;
- a name to be used for the registration in the DNS of the global IPv6 address assigned to the client side of the tunnel;
- the client function (i.e., standalone host or router).

Moreover, if the client machine is an IPv6 router willing to provide connectivity to several IPv6 hosts, the client should be asked also to provide some information about the amount of IPv6 addresses required. This allows the TB to allocate the client an IPv6 prefix that fits its needs instead of a single IPv6 address.

The TB manages the client requests as follows:

- it first designates (e.g., according to some load sharing criteria defined by the TB administrator) a Tunnel Server to be used as the actual tunnel end-point at the network side;
- it chooses the IPv6 prefix to be allocated to the client; the prefix length can be anything between 0 and 128, most common values being 48 (site prefix), 64 (subnet prefix) or 128 (host prefix);
- it fixes a lifetime for the tunnel;
- it automatically registers in the DNS the global IPv6 addresses assigned to the tunnel end-points;
- it configures the server side of the tunnel;

- it notifies the relevant configuration information to the client, including tunnel parameters and DNS names.

After the above configuration steps have been carried out (including the configuration of the client), the IPv6 over IPv4 tunnel between the client host/router and the selected TS is up and working, thus allowing the tunnel broker user to get access to the 6bone or any other IPv6 network the TS is connected to.

2.4 IPv6 address assignment

The IPv6 addresses assigned to both sides of each tunnel must be global IPv6 addresses belonging to the IPv6 addressing space managed by the TB.

The lifetime of these IPv6 addresses should be relatively long and potentially longer than the lifetime of the IPv4 connection of the user. This is to allow the client to get semipermanent IPv6 addresses and associated DNS names even though it is connected to the Internet via a dial-up link and gets dynamically assigned IPv4 addresses through DHCP.

2.5 Tunnel management

Active tunnels consume precious resources on the tunnel servers in terms of memory and processing time. For this reason it is advisable to keep the number of unused tunnels as small as possible deploying a well designed tunnel management mechanism.

Each IPv6 over IPv4 tunnel created by the TB should at least be assigned a lifetime and removed after its expiration unless an explicit lifetime extension request is submitted by the client.

Obviously this is not an optimal solution especially for users accessing the Internet through short-lived and dynamically addressed IPv4 connections (e.g., dial-up links). In this case a newly established tunnel is likely to be used just for a short time and then never again, in that every time the user reconnects he gets a new IPv4 address and is therefore obliged either to set-up a new tunnel or to update the configuration of the previous one. In such a situation a more effective tunnel management may be achieved by having the TS periodically deliver to the TB IPv6 traffic and reachability statistics for every active tunnel. In this way, the TB can enforce a tunnel deletion after a period of inactivity without waiting for the expiration of the related lifetime which can be relatively longer (e.g., several days).

Another solution may be to implement some kind of tunnel management protocol or keep-alive mechanism between the client and the TS (or between the client and the TB) so that each tunnel can be immediately released after the user disconnects (e.g., removing his tunnel end-point or tearing down his IPv4 connection to the Internet). The drawback of this policy mechanism is that it also requires a software upgrade on the client machine in order to add support for the ad-hoc keep-alive mechanism described above.

Moreover, keeping track of the tunnel configuration even after the user has disconnected from the IPv4 Internet may be worth the extra cost. In this way, in fact, when the user reconnects to the Internet, possibly using a different IPv4 address, he could just restart the tunnel by getting in touch with the TB again. The TB could then order a TS to re-create the tunnel using the new IPv4 address of the client but reusing the previously allocated IPv6 addresses. That way, the client could preserve a nearly permanent (static) IPv6 address even though its IPv4 address is dynamic. It could also preserve the associated DNS name.

2.6 Interactions between client, TB, TS and DNS

As previously stated, the definition of a specific set of protocols and procedures to be used for the communication among the various entities in the Tunnel Broker architecture is outside of the scope of the present framework document. Nevertheless, in the reminder of this section some viable technical alternatives to support client-TB, TB-TS and TB-DNS interactions are briefly described in order to help future implementation efforts or standardization initiatives.

The interaction between the TB and the user could be based on http. For example the user could provide the relevant configuration information (i.e., the IPv4 address of the client side of the tunnel, etc.) by just filling up some forms on a Web server running on the TB. As a result the server could respond with an html page stating that the server end-point of the tunnel is configured and displaying all the relevant tunnel information.

After that, the most trivial approach would be to leave the user to configure the client end-point of the tunnel on his own. However, it should be highly valuable to support a mechanism to automate this procedure as much as possible.

Several options may be envisaged to assist the Tunnel Broker user in the configuration of his dual-stack equipment. The simplest option is that the TB could just prepare personalized activation and de-activation scripts to be run off-line on the client machine to achieve easy set-up of the client side tunnel end-point. This

solution is clearly the easiest to implement and operate in that it does not require any software extension on the client machine. However, it raises several security concerns because it may be difficult for the user to verify that previously downloaded scripts do not perform illegal or dangerous operations once executed.

The above described security issues could be elegantly overcome by defining a new MIME (Multipurpose Internet Mail Extension) content-type (e.g., application/tunnel) [4,5] to be used by the TB to deliver the tunnel parameters to the client. In this case, there must be a dedicated agent running on the client to process this information and actually set-up the tunnel end-point on behalf of the user. This is a very attractive approach which is worth envisaging. In particular, the definition of the new content-type might be the subject of a future ad-hoc document.

Several options are available also to achieve proper interaction between the broker and the Tunnel Servers. For example a set of simple RSH commands over IPsec could be used for this purpose. Another alternative could be to use SNMP or to adopt any other network management solution.

Finally, the Dynamic DNS Update protocol [6] should be used for automatic DNS update (i.e., to add or delete AAAA, A6 and PTR records from the DNS zone reserved for Tunnel Broker users) controlled by the TB. A simple alternative would be for the TB to use a small set of RSH commands to dynamically update the direct and inverse databases on the authoritative DNS server for the Tunnel Broker users zone (e.g. broker.isp-name.com).

2.7 Open issues

Real usage of the TB service may require the introduction of accounting/billing functions.

3. Known limitations

This mechanism may not work if the user is using private IPv4 addresses behind a NAT box.

4. Use of the tunnel broker concept in other areas

The Tunnel Broker approach might be efficiently exploited also to automatically set-up and manage any other kind of tunnel, such as a multicast tunnel (e.g., used to interconnect multicast islands within the unicast Internet) or an IPsec tunnel.

Moreover, the idea of deploying a dedicated access-control server, like the TB, to securely authorize and assist users that want to gain access to an IPv6 network might prove useful also to enhance other transition mechanisms. For example it would be possible to exploit a similar approach within the 6to4 model to achieve easy relay discovery. This would make life easier for early 6to4 adopters but would also allow the ISPs to better control the usage of their 6to4 relay facilities (e.g., setting up appropriate usage policies).

5. Security Considerations

All the interactions between the functional elements of the proposed architecture need to be secured:

- the interaction between the client and TB;
- the interaction between the TB and the Tunnel Server;
- the interaction between the TB and the DNS.

The security techniques adopted for each of the required interactions is dependent on the implementation choices.

For the client-TB interaction, the usage of http allows the exploitation of widely adopted security features, such as SSL (Secure Socket Layer) [7], to encrypt data sent to and downloaded from the web server. This also makes it possible to rely on a simple "username" and "password" authentication procedure and on existing AAA facilities (e.g., RADIUS) to enforce access-control.

For the TB-TS interaction secure SNMP could be adopted [8,9,10]. If the dynamic DNS update procedure is used for the TB-DNS interaction, the security issues are the same as discussed in [11]. Otherwise, if a simpler approach based on RSH commands is used, standard IPsec mechanisms can be applied [12].

Furthermore, if the configuration of the client is achieved running scripts provided by the TB, these scripts must be executed with enough privileges to manage network interfaces, such as an administrator/root role. This can be dangerous and should be considered only for early implementations of the Tunnel Broker approach. Transferring tunnel configuration parameters in a MIME type over https is a more secure approach.

In addition a loss of confidentiality may occur whenever a dial-up user disconnects from the Internet without tearing down the tunnel previously established through the TB. In fact, the TS keeps tunneling the IPv6 traffic addressed to that user to his old IPv4

address regardless of the fact that in the meanwhile that IPv4 address could have been dynamically assigned to another subscriber of the same dial-up ISP. This problem could be solved by implementing on every tunnel the keep-alive mechanism outlined in section 2.5 thus allowing the TB to immediately stop IPv6 traffic forwarding towards disconnected users.

Finally TBs must implement protections against denial of service attacks which may occur whenever a malicious user exhausts all the resources available on the tunnels server by asking for a lot of tunnels to be established altogether. A possible protection against this attack could be achieved by administratively limiting the number of tunnels that a single user is allowed to set-up at the same time.

6. Acknowledgments

Some of the ideas refining the tunnel broker model came from discussion with Perry Metzger and Marc Blanchet.

7. References

[1] Gilligan, R. and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers", RFC 1933, April 1996.

[2] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529, March 1999.

[3] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds without Explicit Tunnels", Work in Progress.

[4] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, RFC 2045, November 1996.

[5] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[6] Vixie, P., Editor, Thomson, T., Rekhter, Y. and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[7] Guttman, E., Leong, L. and G. Malkin, "Users' Security Handbook", FYI 34, RFC 2504, February 1999.

[8] Wijnen, B., Harrington, D. and R. Presuhn, "An Architecture for Describing SNMP Management Frameworks", RFC 2571, April 1999.

[9] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2574, April 1999.

[10] Wijnen, B., Presuhn, R. and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 2575, April 1999.

[11] Eastlake, D., "Secure Domain Name System Dynamic Update", RFC 2137, April 1997.

[12] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

8. Authors' Addresses

Alain Durand
SUN Microsystems, Inc
901 San Antonio Road
MPK17-202
Palo Alto, CA 94303-4900
USA

Phone: +1 650 786 7503
EMail: Alain.Durand@sun.com

Paolo Fasano S.p.A.
CSELT S.p.A.
Switching and Network Services - Networking
via G. Reiss Romoli, 274
10148 TORINO
Italy

Phone: +39 011 2285071
EMail: paolo.fasano@cselt.it

Ivano Guardini
CSELT S.p.A.
Switching and Network Services - Networking
via G. Reiss Romoli, 274
10148 TORINO
Italy

Phone: +39 011 2285424
EMail: ivano.guardini@cselt.it

Domenico Lento
TIM
Business Unit Project Management
via Orsini, 9
90100 Palermo
Italy

Phone: +39 091 7583243
EMail: dlento@mail.tim.it

9. Full Copyright Statement

Acknowledgement

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