Internet Engineering Task Force
Internet-Draft
Expires: July 28, 2005

J. Palet
M. Diaz
Consulintel
P. Savola
CSC/FUNET
January 24, 2005

Analysis of IPv6 Tunnel End-point Discovery Mechanisms
draft-palet-v6ops-tun-auto-disc-03.txt

Status of this Memo

Copyright Notice

Abstract

To be able to automate setting up IPv6-in-IPv4 tunnels, it is important to be able to automatically determine the tunnel end-point for the tunneling mechanism. This memo presents a short analysis and conclusions on the different approaches for discovering the IPv6

tunnel endpoint on a node.

Table of Contents

1. Introduction

It is important to make setting up IPv6 connectivity simpler, so that IPv6-ignorant novice users can get the benefit of IPv6 transparently, without the user even having to know that IPv6 connectivity has been obtained.

While this has become possible with Teredo and 6to4, they do not provide well for managed infrastructure, where the addresses come from the service provider's prefix.

This document presents a short analysis and conclusions for different options to automatically determine the IPv6-in-IPv4 (with or without UDP encapsulation) tunnel end-points to a tunnel server, so that the set up of tunnels could be automated.

Note that the other end-point ("tunnel server") typically also needs to have a means to configure the client's end-point, but that is assumed to be solved by the tunnel server mechanism [1][7], and beyond the scope of this memo.

Some form of automatic discovery already exists in some already specified mechanisms; the generic discovery is out of scope, but we contrast the approaches to already-deployed methods when appropriate. For example, 6to4 [2] uses global anycast [3] and/or vendor's branch of DNS, Teredo [8] uses vendor's branch of DNS, and ISATAP [9] uses search-path -prefixed DNS.

First, we look at manual configuration, and why it is not considered sufficient.

1.1 Manual Configuration

Users typically expect to be able to manually configure the tunnel endpoint information, and the implementations obviously should allow that.

Some implementations may also provide some default values (e.g., using a vendor branch of DNS, as described in Section 3.2). This is a non-interoperability issue, and may also be a good idea.

Often the ISPs also provide CD-ROMs or other material to the (non-knowledgeable) users which will automatically reset the network connectivity settings to the values used by the ISP. Such configuration could also include the tunnel endpoint if the ISP would like to roll it out to every customer.

This raises the question whether it is strictly required to have an

automatic discovery process, rather than wait and rely on the ISPs to do a massive roll-out.

We do not consider manual configuration sufficient for two main reasons (XXX: feel free to send some more..):

1. Many deployments are expected to happen at pre-production stage, and those service providers would not yet be ready to integrate the configuration in their CD-ROM etc. material.

2. The CD-ROM materials have mainly been targeted to the users who set up their (IPv4) connectivity which either requires software or out-of-band configuration. There is no reason to require out-of-band configuration if the discovery could be done in a feasible manner.

2. Applicability of Tunnel Endpoint Discovery

2.1 Scope of Tunnel Endpoint Discovery

There are three main areas of applicability for tunnel endpoint discovery:

1. No discovery: always assume the client pre-configures the endpoint information.

2. Discovery of the end-point at the "care-of" ISP only; that is, discovery is only supported inside the ISP of the network the user plugs into.

3. Discovery of the endpoint everywhere; contrast to the use of 6to4 anycast address. This is very problematic administratively, financially and technically, because the IPv6 prefix is not provider-independent as with 6to4.

We decree 3) out of scope for this study; this is too extensive a problem to be solved here. Therefore, we concentrate on 2).

In addition to automatic discovery, the implementations should naturally provide a manual configuration option. The manual configuration could be used to override the automatic discovery process or to configure a tunnel server at "home ISP" which the discovery would not find if the user is visiting a network where no tunnel server exists (compare to configuring MIPv6 Home Agent).

When there are multiple inputs to which tunnel server to use, implementations will have to make a policy decision which one to

pick; this is rocket science, and some implementations (e.g., Microsoft 6to4/ISATAP) already something like this. The main options are:

a. First try local discovery, if it fails, try manual config,

b. Manual config first, then try local discovery, or

c. Only local discovery or only manual config.

2.2 Assumptions about Network Topologies

The following assumptions about network topologies apply to the discovery process:

1. The CPE device can run either in bridged or routed mode.

2. In routed mode, the router typically is doing NAT with private IP addresses. The router is also a DHCPv4 server, so DHCPv4 tunnel endpoint option is not relayed to the user.

3. The router may also inject its own DNS search string (e.g., "home", "lan") instead relaying the one received from the ISP, though how often that happens is unknown.

4. The user may also have deployed NAT boxes of his/her own.

3. Analysis of Solutions

Several possible solutions to discovering the tunnel end-point can be imagined; this section describes them in detail.

3.1 Anycast-based Solutions

An "anycast" (shared-unicast by some terminology: see [10]) address identifies a group of hosts, usually server hosts. When a client sends a datagram to a shared-unicast address, it is delivered to one of the shared-unicast servers based on the routing topology and metrics.

There are two possible ways of using "anycast": as a global service (where a shared-unicast prefix is the same for everyone, and advertised in the Inter-domain routing) or as a local service, where the service provider is sharing one of its own addresses on multiple nodes for example for load-balancing or redundancy reasons. As local anycast is invisible to the users, it is not further discussed here.

A packet to a shared-unicast address may end up being delivered to more than one node. In addition, there is no guarantee that two consecutive datagrams sent from the same host towards the same shared-unicast address are going to be delivered to the same node. However, when the routing topology is stable and metrics are well-designed, the packets are regularly delivered to the same nodes. Operational issues relating to managemeng of anycast services have been described in [4].

A global anycast address could be leveraged in two fundamentally different ways:

1. Use the anycast address only for the initial handshake, to establish a stable unicast address of the end-point (and possibly to perform some initial negotiation, e.g., nonces). All the subsequent packets are sent to the unicast address which is included in the payload of the reply. An example of such use is in [5].

2. Use the anycast address for all the communications (e.g., as with 6to4).

The former approach is much more suitable in this situation as the IPv6 address/prefix of the tunnel service depends on the operator of the service. The cost is at least one additional roundtrip.

The failure modes of the initial handshake anycast are described in Appendix A.

The advantages of the anycast approach are:

- Works well also in the presence of NATs and does not require any other components like DNS or DHCP.

- The routing stability and leaks are not a major concern if the anycast address is only used for initial discovery. In other words, the worst that could happen is that if the initial discovery does not work correctly at the moment, and the user is either cannot get any service or directed to a tunnel server which does not offer any service.

The drawbacks are:

- Setting up an internal anycast route advertisement (e.g., in an enterprise) is likely a little bit more difficult than adding a name in the DNS or configuring DHCP.

- The use of non-local prefixes may require changes in firewall IP

prefix, access lists, etc.

- The failure modes are a bit more complex than, e.g., just looking up a domain name, as one will have to send 1-2 packets to the anycast address to see if a working tunnel server is found or not.

In summary, from the automatic methods, a global anycast based solution, using the address only for initial handshake, seems like a very promising approach, but has routing operations issues which need to be considered.

3.2 DNS-based Solutions

As DNS is globally deployed and easy to use, it could provide a means for discovering the end-point address, either based on the forward or reverse tree.

There are roughly four kinds of different approaches:

1. "(forward) global name": the systems look up a globally unique name, like www.tunnel-server.net which would point to the global anycast address. This is not considered further as this does not solve any problem in itself, because the clients could have been configured with IP address instead.

2. "(forward) vendor branch": the operating system vendors may provide a DNS record which is looked up (contrast to "6to4.windows.microsoft.com."), giving the vendor some control over already deployed systems. This could in practice only be used to configure the global anycast address, because the authors don't expect the vendors would provide a tunnel server to all the customers all over the world. Therefore this approach is not considered further.

3. "(forward) prefixing the search path" [6]: one could look up a service-specific special string, like "_tunnel-server", appended by the DNS search path, e.g., "isp.example.com", resulting in a query of "_tunnel-server.isp.example.com".

4. "(reverse) querying the IP addresses for TEP information": for example looking up a special record for the assigned IP address.

It is also a question where to store the information; the main suggestions have been at least A/CNAME, SRV, NAPTR and new type of records. We first discuss these options.

Approaches 3) and 4) are a bit more complicated and have different tradeoffs, so they are elaborated after looking at how to store the

information.

3.2.1 Storing the TEP Information

Forward-tree global name and service branch would obviously use A/CNAME records.

Forward-tree search path prefixing could use A/CNAME records. Architecturally a bit "cleaner" approach would be to use SRV records, which also provide a bit more fine-grained means for load distribution. NAPTR provide even more extended load distribution, but it is not clear what the benefit would be. A new RR could also be defined, but there seems to be no particular reason to do so, and a lot of drawbacks in the process.

Reverse-tree IP address lookup would likely have to define a new record type.

The main benefit of using A/CNAME records would be the applications can use simple getaddrinfo() lookups, instead of having to write their own or use a non-standardized DNS record lookup functions (e.g., getrrsetbyname).

3.2.2 Prefixing the DNS Search Path

Prefixing the search path bears a bit more analysis; we discussed how to store the information above, and now look at where to store the records (i.e., the prefix to use, and what to do with the conflicts).

This approach makes two assumptions; there are cases of both when these do not hold:

1. There is a decent mapping between the DNS hierarchy and the routing topology.

2. The information propagated to the end nodes in the "DNS search path" is relevant to figure out the domain name of the whoever is providing the tunnel service.

The main advantages of the solution are:

- The discovery process is simple, and is already used for example by ISATAP (but without NATs in the middle).

- Adding the service is very simple, as the ISP is only required to add one address in their DNS zone.

However, the main drawbacks of this approach are:

- Some routers and middleboxes may not propagate the search path, but to insert their own, and this approach does not work in that situation. It is not clear how widespread this is. (In case the NAT would insert a new search path, e.g., "lan", they also should have be authorative for the zone, otherwise the root servers get bombarded by lookups.)

- In some cases in global enterprises, forward DNS may not map as well to the physical topology as IP addresses through reverse DNS would. (NB: then ISATAP would have the same issue as well.)

In summary, it is questionable how well prefixing the search path works under these circumstances, and in particular how common propagating (or not) the search path is.

3.2.3 IP-address Query from Reverse DNS

The node might also try to find out its tunnel endpoint information by querying its own IP address in the reverse DNS for a certain DNS record type.

The name needs to be stored somewhere. The options are basically queries like (for IP address 192.0.2.1):

1. "QNAME=_tunnel.1.2.0.192.in-addr.arpa. QTYPE=A"

2. "QNAME=1.2.0.192.in-addr.arpa. QTYPE=TEP"

In the first case, an arbitrary subname would have to be defined; one could query these for A, PTR, or some other records directly.

In the second case, where the PTR records for the name might already exist one should probably use a new record type, though something like NAPTR has been used in private experiments.

Advantages:

- IP address maps very well to the topology.

- Querying a new record type is an architecturally relatively clean approach.

Drawbacks:

- Does not work well with NAT; would require that the ISPs prepopulate all the private address space records.

- Major management problems. Wildcards cannot be used because they

would be in the middle of the QNAME or would match other records (such as PTR) because wildcards are QTYPE-agnostic.

- Reverse DNS is not necessarily managed by those who would like to configure this service.

- A new record type number would be available (for test deployments, interoperability etc.) only after an RFC has been published.

In some cases, the reverse records are generated by scripts which could be modified to also add these records. However, the presence of such scripts and the ability to modify them cannot be assumed.

In short, storing information in the reverse DNS does not look like a good approach either.

3.3 DHCP-based Solutions

In most situations, the users receive the IPv4 information from an IPv4 DHCP server. Consequently, one of the parameters to be provided by the DHCP server could be the tunnel end-point address, e.g., as described in [11].

This approach has several drawbacks:

- It requires standardizing new parameters/options on this protocol and also upgrading the DHCP client/server implementations to support this feature.

- It will not work if DHCP client is not used, e.g., in many dial-up scenarios, where only PPP is used; DHCP is not used in some (advanced) xDSL setups which use static routing. Also, some managed networks do not use DHCP. Still, in many cases, DHCP is used between a customer and the ISP.

- If a router is providing local DHCP information (e.g., an ADSL router), the tunnel end-point information would have to be automatically "proxied" to the "local DHCP", or manually configured on the router to propagate to the hosts in the case that the router is not activating the tunnel itself.

- It requires manual configuration/update of the ISP's DHCP servers when there are changes to the tunnel end-points, similar to updating DNS, NTP, etc., server information.

In short, DHCP-based solutions seem unacceptable because a NAT/router does not automatically pass the information to the nodes in an "opaque" manner.

3.4 PPP-based Solutions

In the case of PPP-like connections, specific PPP parameters could be passed to the clients, as part of the AAA signaling process.

This solution has the same drawbacks as DHCP-based solution. Further, there has been resistance to making extensions to PPP (e.g., passing IPv6 prefix options), so it is an open question whether this information could be passed as a standardized PPP option at all.

3.5 SLP-based Solutions

The Service Location Protocol [12] provides a framework for the discovery and selection of network services. Tunnel-End-Point for IPv6 tunnels could be defined as a network service which will have also assigned a specific service name.

SLP has a number of drawbacks:

- SLP is not really widely implemented or deployed.
- It requires multicast infrastructure or the additional deployment of Directory Agents (DA) for Service Agent (SA) discovery.
- If DA is deployed and the network has not multicast support, some way for discovery the DA is required. DHCP could be used [12] but this has the same issues why a DHCP-based approach is not sufficient (in Section 3.3).
- It requires the implementation of a User Agent (UA) on the client's host. This is neither always possible nor feasible.
- It can not offer any kind of load-balancing if more than one TEP is deployed.

In short, SLP seems to come awfully short in matching the requirements for the solution.

3.6 Combined Solutions

Many solutions can be combined with each other, but because the clients and servers must have a minimum mandatory-to-implement mechanism, it is better if only one externally visible mechanism can be used.

Manual configuration override option is of course a good addition to any discovery mechanism.

Anycasting a locally determined TEP address (e.g., through DNS or DHCP) is a useful technique for load balancing purposes. This is invisible to the clients.

As has been experienced with 6to4, it might be possible to use a vendor's DNS branch to configure a global anycast address, but this the former requires no interoperability, so it's an implementation's internal matter.

4. Conclusions

Manual configuration should always be provided, but the question is whether the configuration distributed by ISPs (e.g., in CD-ROMs etc) is sufficient.

Global anycast for initial discovery looks like a promising solution, but has routing operations issues which need to be considered.

Both DNS forward and reverse based solutions suffer from various problems when a NAT/router is present.

Centralized brokers are a non-starter.

DHCP and PPP do not seem to be usable due to restrictions on environment where they work. SLP is not sufficiently deployed, implemented or otherwise feasible.

The following table summarizes the pros/cons of different approaches presented in this memo.

	Pros	Cons
Anycast	***	*
Centralized Broker	-	***
Forward DNS w/ Prefixing	**	**
Reverse DNS	**	***
DHCP	*	***
PPP	-	***
SLP	-	***

Qualification of pros/cons:

- \- : None

- * : Few

** : Medium

*** : High

5. Security Considerations

If the tunnel end-point discovery is done in an insecure fashion, so that an attacker could influence the discovery process, the attacker could be able to hijack all the IPv6 communications. This must be kept in mind when analyzing the different discovery solutions, and spelled-out explicitly in the requirements, if the threats are to be mitigated in tunneling mechanisms somehow (e.g., using a return routability procedures).

In particular, the potential weaknesses of DNS bear some consideration.

6. IANA Considerations

This document requests no action for IANA.

[[note to RFC-editor: this section can be removed upon publication.]]

7. Acknowledgements

This memo was written as a consequence of experience using IPv6 when traveling, number of talks during IETF meetings and specially the work with the unmanaged, ISP and enterprise v6ops design teams.

The authors would also like to acknowledge inputs from Carl Williams, Brian Carpenter, Jeroen Massar, Alain Durand, Tim Chown, and Florent Parent. This work has been partially funded by the European Commission under the Euro6IX project.

8. References

8.1 Normative References

[1] Parent, F., "Goals for Registered Assisted Tunneling", Internet-Draft draft-ietf-v6ops-assisted-tunneling-requirements-01 , October 2004.

[2] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[3] Huitema, C., "An Anycast Prefix for 6to4 Relay Routers", RFC 3068, June 2001.

[4] Lindqvist, K. and J. Abley, "Operation of Anycast Services", Internet-Draft draft-kurtis-anycast-bcp-00, October 2004.

[5] Thaler, D. and L. Vicisano, "IPv4 Automatic Multicast Without Explicit Tunnels (AMT)", Internet-Draft draft-ietf-mboned-auto-multicast-03, October 2004.

[6] Faltstrom, P. and R. Austein, "Design Choices When Expanding DNS", Internet-Draft draft-iab-dns-choices-00, October 2004.

8.2 Informative References

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

[8] Huitema, C., "Teredo: Tunneling IPv6 over UDP through NATs", Internet-Draft draft-huitema-v6ops-teredo-04, January 2005.

[9] Templin, F., Gleeson, T., Talwar, M. and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", Internet-Draft draft-ietf-ngtrans-isatap-23, December 2004.

[10] Hagino, J. and K. Ettican, "An analysis of IPv6 anycast", Internet-Draft draft-ietf-ipngwg-ipv6-anycast-analysis-02, June 2003.

[11] Kim, P. and S. Park, "DHCP Option for Configuring IPv6-in-IPv4 Tunnels", Internet-Draft draft-daniel-dhc-ipv6in4-opt-05, October 2004.

[12] Guttman, E., Perkins, C., Veizades, J. and M. Day, "Service Location Protocol, Version 2", RFC 2608, June 1999.

[13] Brisco, T., "DNS Support for Load Balancing", RFC 1794, April 1995.

[14] Johnson, D., Perkins, C. and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.

[15] Durand, A. and S. Yamamoto, "Service Discovery using NAPTR records in DNS", Internet-Draft draft-yamamoto-naptr-service-discovery-00, October 2004.

Authors' Addresses

Jordi Palet Martinez
Consulintel
San Jose Artesano, 1
Alcobendas - Madrid
E-28108 - Spain

Phone: +34 91 151 81 99
Fax: +34 91 151 81 98
Email: jordi.palet@consulintel.es

Miguel Angel Diaz Fernandez
Consulintel
San Jose Artesano, 1
Alcobendas - Madrid
E-28108 - Spain

Phone: +34 91 151 81 99
Fax: +34 91 151 81 98
Email: miguelangel.diaz@consulintel.es

Pekka Savola
CSC/FUNET
Espoo
Finland

Email: psavola@funet.fi

Appendix A. More Discussion of Anycast Discovery

We do a little bit of more extensive analysis of anycast-based solution here to get a better understanding of its operational properties.

At this point, we only discuss the different failure modes of initial handshake anycast, and see that these can be solved with robust specification and implementation.

a. A new server starts advertising the address but is refusing to serve the users: established tunnels continue to work, new tunnels cannot be established; tracerouting to the server identifies where the culprit is.

b. The current anycast server goes down: tunnels established to its unicast address go down after the event is detected. New tunnels

use the next anycast server (if available) or no server at all (in which case the tunnels may get re-created when the server comes back up).

c. There is no server in the local ISP, and the anycast prefix has been filtered out or does not exist globally: the initial discovery follows the default route, and either gets discarded (and the discovery process notices this after a timeout) or a router returns a network/host unreachable ICMP error message.

d. There is no server locally, but someone else is advertising it here: if the service works, that's OK (though if the service works but is bad quality or 100's of milliseconds away is undesirable; robust implementations may check the RTT). If the server fails to serve, that's also OK -- the discovery process has to be robust against this.

Appendix B. Centralized Broker-based Solutions

This solution is described in the appendix because it does not actually solve the discovery problem, but has been proposed on the assumption that it would.

Inside a single administrative domain, it would also be possible to deploy a centralized server or a "broker" knowing the status of all the associated end-points. Furthermore, it could redirect the users to the correct end-points. This mechanism would still need another complementary approach to actually discover the centralized broker.

This approach is highly assumptive of the tunneling set-up mechanism, and likely requires the implementation of lengthy redirection or negotiation features which do not work well through a NAT.

Applying a centralized model over multiple administrative domains, e.g., having a single server for the whole Internet, would be administratively and management-wise unfeasible.

A global centralized broker is completely unfeasible. The only benefit of a local broker is the ability to perform more fine-grained load balancing or policing, and does not solve the actual problem, because the same methods need to be applied to discover the centralized broker.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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