Network Working Group
Internet-Draft
Expires: August 9, 2004

M. Blanchet
Hexago

February 9, 2004

IPv6 Tunnel Broker with the Tunnel Setup Protocol(TSP)
draft-blanchet-v6ops-tunnelbroker-tsp-00

Status of this Memo

Copyright Notice

Abstract

A tunnel broker with the Tunnel Setup Protocol(TSP) enables the establishment of tunnels of various inner protocols, such as IPv6 or IPv4, inside various outer protocols packets, such as IPv4, IPv6 or UDP over IPv4 for IPv4 NAT traversal. The control protocol (TSP) is used by the tunnel client to negociate the tunnel with the broker. The negociation involves authentication, authorization, tunnel information such as IP addresses, prefixes when the client is a router, DNS information such as the NS for the inverse zone corresponding to the delegated prefix, etc. Some parameters may be proposed by the broker, such as the transport over UDP IPv4 where an IPv4 NAT is found in the path between the client and the broker. A mobile node implementing TSP can be connected to both IPv4 and IPv6

networks whether he is on IPv4, IPv4 behind a NAT or on IPv6. A tunnel broker may terminate the tunnels on remote tunnel servers or on itself. This document describes the TSP protocol within the model of the tunnel broker [3].

Table of Contents

1. Introduction

This document first describes the TSP framework as well as the different profiles used. It then describes the applicability of TSP in different environments, some were described in the v6ops scenario documents.

2. Description of the TSP framework

Tunnel Setup Protocol (TSP) is a control/signaling protocol to setup tunnel parameters between two tunnel end-points. TSP is implemented as a tiny client code in the requesting tunnel end-point. The other end-point is the TSP server. TSP uses XML basic messaging over TCP or UDP. The use of XML gives extensibility and easy option processing.

Inside a session, TSP can negociate between the two tunnel end-points:

- authentication of the users, using any kind of authentication mechanism(through SASL [1]) including anonymous
- IPv6 over IPv4 tunnels
- IPv4 over IPv6 tunnels
- IPv6 over UDP-IPv4 tunnels, when IPv4 NAT are in the path between the two endpoints

- o IP address allocation for the tunnel endpoints
- o IPv6 prefix assignment when the client is a router and has a network behind
- o DNS delegation of the inverse tree, based on the ipv6 prefix assignment
- o DNS registration of the end point.
- o Routing protocols

The TSP connexion can be established between two nodes, where each node can control a tunnel end-point. In this context, it is possible to have up to 4 parties involved: 1- the tsp client, 2- controlling the requesting tunnel end-point, 3- the tsp server, 4- controlling the receiving tunnel end-point. 1,3 and 4 is the Tunnel Broker model. 1 and 2 can be on the same node, as well as 3 and 4 can be on the same node.

From the point of view of an operating system, TSP is implemented as a client application which is able to configure network parameters of the kernel and operating system.

2.1 NAT Discovery

TSP is also used to discover if a NAT is in the path. In this discovery mode, the client sends a TSP message, containing its source tunnel information (such as its source IPv4 address) and the request for the tunnel over UDP-IPv4 to the TSP server. The TSP server compares the IPv4 source address of the packet with the IPv4 source address in the TSP messaging. If they differ, a NAT was in the path.

If an IPv4 NAT is discovered, then UDP-IPv4 encapsulation of the IPv6 tunnel is used over the same UDP channel used for TSP, which enables the use of the same NAT address-port mapping for both the TSP session and the IPv6 traffic. A keepalive mechanism is also included to keep the NAT mapping constant. If there is no IPv4 NAT in the path as verified by the tunnel broker, then usual IPv6 in IPv4 encapsulation is used.

When the TSP client moves to another network, the same discovery process is done. This IPv4 NAT discovery builds the most effective tunnel for all cases, including in a dynamic situation where the client moves. On the IPv6 layer, if the client have used user authentication, the same IPv6 address and prefix are kept and re-established. On the IPv6 layer, there are no mobility seen.

2.2 Any encapsulation

TSP is used to negociate IPv6 over IPv4 tunnels, IPv6 over UDP-IPv4 tunnels and IPv4 over IPv6 tunnels. IPv4 in IPv6 tunnels are used in the Dual Stack Transition Mechanism (DSTM) together with TSP.

2.3 Mobility

When a tunnel endpoint changes its underlying IP address (i.e. change of its IPv4 address when doing IPv6 in IPv4 encapsulation), the keepalive mechanism fail and the TSP client reconnects to the broker to re-establish the tunnel.

3. Advantages of TSP

- A signaling protocol to establish the tunnel: no need to change kernels, routing...
- A signaling protocol flexible and extensible

- one solution to many encapsulation techniques: v6 in v4, v4 in v6, v6 over udp over v4, ...
- prefix assignment
- dns delegation
- routing negociation
- discovery of IPv4 NAT in the path, establishing the most optimized tunnelling technique depending on the discovery.
- mobility of the underlying IP node.

- o two to four tier tunnel broker model
- o stability of the IP address and prefix, enabling applications needing stable address to be deployed and used.
- o Tunnels established by TSP are static tunnels, which are more secure than automated tunnels

4. Protocol Description

4.1 Terminology

Tunnel Broker (TB) In a Tunnel Broker model, the broker is taking charge of all communication between Tunnel Servers (TS) and Tunnel Clients (TC). Tunnel clients query brokers for a tunnel and the broker find a suitable tunnel server, ask the Tunnel server to setup the tunnel and send the tunnel information to the Tunnel Client.

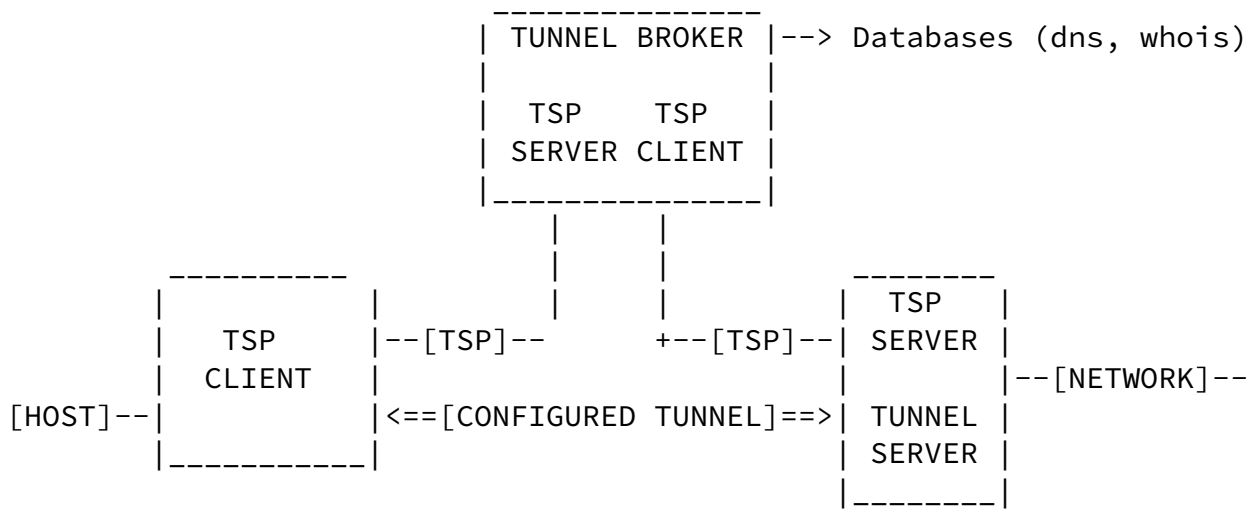
Tunnel Server (TS) Tunnel Servers are providing the specific tunnel service to a Tunnel Client. It can reveive the tunnel request from a Tunnel Broker (as in the Tunnel Broker model) or directly from the Tunnel Client as in the Tunnel Setup Protocol option. The Tunnel Server is the tunnel end-point.

Tunnel Client (TC) The Tunnel Client is the entity that need a tunnel for a particular service or connectivity. A Tunnel Client can be either a host or a router. The tunnel client is the other tunnel end-point.

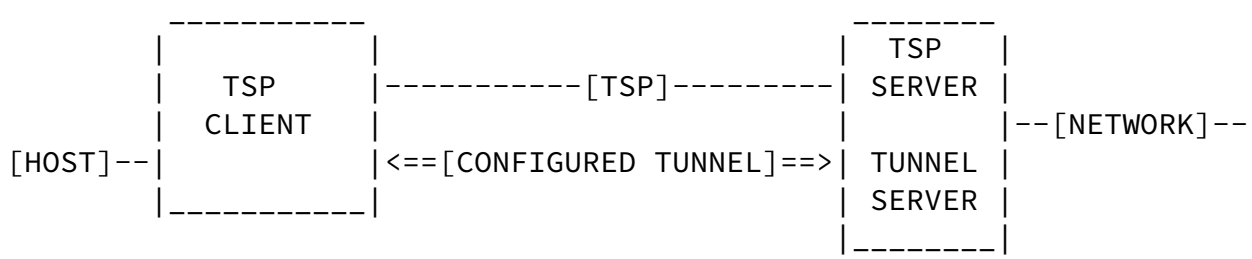
4.2 Topology

The following diagrams describe typical TSP scenarios. The goal is to establish a tunnel between Tunnel client and Tunnel server.

Tunnel Setup Protocol used on Tunnel Broker model



Tunnel Setup Protocol used on Tunnel Server model



4.3 Overview

The Tunnel Setup Protocol has three phases:

Authentication phase: The Authentication phase is when the Tunnel Brokers/Servers advertises their capability to Tunnel Clients and when Tunnel clients authenticate to the server.

Command phase: The command phase is where the client requests or updates a tunnel.

Response phase: The response phase is where the respond to the client

For each command sent by a Tunnel Client their is an expected response by the server.

4.4 Authentication phase

The authentication phase has 3 steps :

- Client's protocol version identification
- Server's capability advertisement
- Client authentication

When a TCP (or UDP-reliable) session is established to a Tunnel Server, the Tunnel Client sends the current protocol version it is supporting. The version number syntax is:

```
VERSION=2.0 CR LF
```

Version 2.0 is the version number of this specification. Version 1.0 was defined in earlier drafts.

If the server doesn't support the protocol version it sends an error message and closes the session. The server can optionally send a server list that may support the protocol version of the client.

Example of a Version not supported (without a server list)

```
-- Successful TCP Connection --
C:VERSION=0.1 CR LF
S:302 Unsupported client version CR LF
-- Connection closed --
```

Example of a Version not supported (with a server list)

```
-- Successful TCP Connection --
C:VERSION=1.1 CR LF
S:1302 Unsupported client version CR LF
  <tunnel action="list" type="broker">
     <broker>
        <address type="ipv4">1.2.3.4</address>
  </broker>
  <broker>
     <address type="dn">ts1.isp1.com</address>
  </broker>
  </tunnel>
-- Connection closed --
```

If the server supports the version sent by the client, then the server sends a list of the capabilities supported for authentication and tunnels.

```
CAPABILITY TUNNEL=V6V4 AUTH=DIGEST-MD5 AUTH=ANONYMOUS CR LF
```

Tunnel types must be registered with IANA and their profiles are defined in other documents. Authentication is done using SASL [1]. Each authentication mechanism must be a registered SASL mechanism. Description of such mechanism is not in the scope of this document.

The Tunnel Client can then choose to close the session if none of the capabilities fits its needs. If the Tunnel Client chooses to continue, it must authenticate itself to the server using one of the adevertised mechanism. If the authentication fails the server sends an error message and closes the session.

Example of failed authentication

```
-- Successful TCP Connection --
C:VERSION=2.0 CR LF
S:CAPABILITY TUNNEL=V6V4 AUTH=DIGEST-MD5 CR LF
C:AUTHENTICATE ANONYMOUS CR LF
S:300 Authentication failed CR LF
```

If the authentication succeed, the server sends a success return code and the protocol enter the Command phase.

Successful authentication

```
-- Successful TCP Connection --
C:VERSION=2.0 CR LF
S:CAPABILITY TUNNEL=V6V4 AUTH=DIGEST-MD5 AUTH=ANONYMOUS CR LF
C:AUTHENTICATE ANONYMOUS CR LF
S:200 Authentication successful CR LF
```

Upon successful authentication the protocol enters the command phase as described in the next section.

4.5 Command phase

The Command phase is where the Tunnel Client send a tunnel request or a tunnel update to the server. In this phase, commands are sent as XML messages. The first line is a "Content-length" directive that tells the size of the following XML message. This makes it easier for protocol implementation to tell when they got the whole XML message. When the server sends a response, the first line is the

"Content-length" directive, the second is the return code and third one is the XML message if any. The size of the response for the "Content-length" directive is the first character of the return code line to the last character of the XML message.

Spaces can be inserted freely.

Example of a command/response sequence

```
-- Successful TCP Connection --
C:VERSION=2.0 CR LF
S:CAPABILITY TUNNEL=V6V4 AUTH=DIGEST-MD5 AUTH=ANONYMOUS CR LF
C:AUTHENTICATE ANONYMOUS CR LF
S:200 Authentication successful CR LF
C: Content-length: 123 CR LF
  <tunnel action="create" type="v6v4">
    <client>
      <address type="ipv4">1.1.1.1</address>
    </client>
  </tunnel> CR LF

S: Content-length: 234 CR LF
   200 OK CR LF
   <tunnel action="info" type="v6v4" lifetime="1440">
    <server>
     <address type="ipv4">206.123.31.114</address>
     <address type="ipv6">3ffe:b00:c18:ffff:0000:0000:0000:0000</addre
    </server>
    <client>
     <address type="ipv4">1.1.1.1</address>
     <address type="ipv6">3ffe:b00:c18:ffff::0000:0000:0000:0001</addr
     <address type="dn">userid.domain</address>
    </client>
   </tunnel> CR LF
-- TCP Connection closed --
```

4.6 XML Messaging

4.6.1 Tunnel

The client uses the tunnel token with an action attribute. Valid actions for this profile are : 'create', 'info' and 'delete'.

The 'create' action is used to request a new tunnel or update an existing tunnel. The 'info' action is used to request current properties of an existing tunnel. The 'delete' action is used to remove an existing tunnel from the server.

The 'tunnel' message contains three elements:

client Client's information

server Server's information

broker List of other server's

4.6.2 Client element

The client element contains 2 elements: 'address' and 'router'. These elements are used to describe the client needs and will be used by the server to create the appropriate tunnel. This is the only element sent by a client.

The 'address' element is used to identify the client IPv4 endpoint of the tunnel. The client MUST send only an IPv4 address to the server. The server will then return the IPv6 address endpoint and domain name inside the 'client' element when the tunnel is created or updated.

Optionaly a client can send a 'router' element to ask for a prefix delegation.

4.6.3 Server element

The 'server' element contains 2 elements: 'address' and 'router'. These elements are used to describe the server's tunnel endpoint. The 'address' element is used to provide both IPv4 and IPv6 addresses of the server's tunnel endpoint, while the 'router' element provides information for the routing method choosen by the client.

4.6.4 broker element

The 'broker' element is used by a server to provide a alternate list of servers to a client in the case where the server is not able to provide the requested tunnel.

The 'broker' element will contain a series of 'address' element.

4.7 Tunnel request examples

This section presents multiple examples of requests.

4.7.1 Host Tunnel request and Reply

A simple tunnel request consist of a 'tunnel' element which contains only an 'address' element

Simple tunnel request made by a client.

```
      -- Successful TCP Connection --
      C:VERSION=1.0 CR LF
      S:CAPABILITY TUNNEL=V6V4 AUTH=ANONYMOUS CR LF
      C:AUTHENTICATE ANONYMOUS CR LF
      S:200 Authentication successful CR LF
      C:Content-length: 123 CR LF
        <tunnel action="create" type="v6v4">
           <client>
               <address type="ipv4">1.1.1.1</address>
           </client>
        </tunnel> CR LF
      S: Content-length: 234 CR LF
         200 OK CR LF
         <tunnel action="info" type="v6v4" lifetime="1440">
           <server>
              <address type="ipv4">206.123.31.114</address>
              <address type="ipv6">3ffe:b00:c18:ffff:0000:0000:0000:0000</ad
           </server>
           <client>
              <address type="ipv4">1.1.1.1</address>
```

```
                <address type="ipv6">3ffe:b00:c18:ffff::0000:0000:0000:0001</a
                <address type="dn">userid.domain</address>
              </client>
            </tunnel> CR LF
```

4.7.2 Router Tunnel request with a /48 prefix delegation, and reply

A tunnel request with prefix consist of a 'tunnel' element which contains 'address' element and a 'router' element.

Tunnel request with prefix and static routes.

```
C: Content-length: 234 CR LF
   <tunnel action="create" type="v6v4">
    <client>
     <address type="ipv4">1.1.1.1</address>
     <router>
      <prefix length="48"/>
      <dns_server>
       <address type="ipv4">2.3.4.5</address>
       <address type="ipv4">2.3.4.6</address>
       <address type="ipv6">3ffe:0c00::1</address>
      </dns_server>
     </router>
    </client>
```



```
   </tunnel> CR LF
S: Content-length: 234 CR LF
   200 OK CR LF
   <tunnel action="info" type="v6v4" lifetime="1440">
    <server>
     <address type="ipv4">206.123.31.114</address>
     <address type="ipv6">3ffe:b00:c18:ffff:0000:0000:0000:0000</address>
    </server>
    <client>
     <address type="ipv4">1.1.1.1</address>
     <address type="ipv6">3ffe:b00:c18:ffff::0000:0000:0000:0001</address>
     <address type="dn">userid.domain</address>
     <router>
      <prefix length="48">3ffe:0b00:c18:1234::</prefix>
      <dns_server>
```

```
        <address type="ipv4">2.3.4.5</address>
        <address type="ipv4">2.3.4.6</address>
        <address type="ipv6">3ffe:0c00::1</address>
       </dns_server>
      </router>
     </client>
    </tunnel> CR LF
```

4.7.3 IPv4 in IPv6 tunnel request

Tunnel type is set to 'v4v6'.

Simple tunnel request made by a client.

```
          -- Successful TCP Connection --
          C:VERSION=1.0 CR LF
          S:CAPABILITY TUNNEL=V4V6 AUTH=DIGEST-MD5 AUTH=ANONYMOUS CR LF
          C:AUTHENTICATE ANONYMOUS CR LF
          S:OK Authentication successful CR LF
          C:Content-length: 228 CR LF
            <tunnel action="create" type="v4v6">
               <client>
                   <address
type="ipv6">fe80:0000:0000:0000:0000:0000:0000:0001</address>
                   <address
type="ipv6">3ffe:0b00:0c18:ffff:0000:0000:0000:0001</address>
               </client>
            </tunnel> CR LF
```

If the allocation request is accepted, the broker will acknowledge the allocation to the client by sending a 'tunnel' element with the

attribute 'action' set to 'info', 'type' set to 'v4v6' and the 'lifetime' attribute set to the period of validity or lease time of the allocation. The 'tunnel' element contains 'server' and 'client' elements.

Server response

```
            S: Content-length: 370 CR LF
               200 OK CR LF
               <tunnel action="info" type="v4v6" lifetime="1440">
                 <server>
                    <address type="ipv4" length="30">206.123.31.2</address>
                    <address type="ipv6">3ffe:b00:c18:ffff:0000:0000:0000:0002
                 </server>
                 <client>
                    <address type="ipv4" length="30">206.123.31.1</address>
                    <address
   type="ipv6">3ffe:b00:c18:ffff::0000:0000:0000:0001</address>
                 </client>
               </tunnel> CR LF
```


4.7.4 NAT Traversal tunnel request

```
-- Successful TCP Connection --
C:VERSION=1.1 CR LF
S:CAPABILITY TUNNEL=V6V4 TUNNEL=V6UDPV4 AUTH=DIGEST-MD5 CR LF
C:AUTHENTICATE ... CR LF
S:200 Authentication successful CR LF
C:Content-length: ... CR LF
  <tunnel action="create" type="v6anyv4">
     <client>
         <address type="ipv4">10.1.1.1</address>
     </client>
  </tunnel> CR LF
S: Content-length: ... CR LF
   200 OK CR LF
   <tunnel action="info" type="v6udpv4" lifetime="1440">
     <server>
        <address type="ipv4">206.123.31.114</address>
        <address type="ipv6">3ffe:b00:c18:ffff:0000:0000:0000:0000</addres
     </server>
     <client>
        <address type="ipv4">10.1.1.1</address>
        <address type="ipv6">3ffe:b00:c18:ffff::0000:0000:0000:0001</addre
     </client>
   </tunnel> CR LF
```

5. Applicability of TSP in Different Environments

This section describes the applicability of TSP in different environments.

5.1 Applicability of TSP in Provider Networks with Enterprise Customers

In a provider network where IPv4 is dominant, a tunnelled infrastructure can be used to provider IPv6 services to the enterprise customers, before a full IPv6 native infrastructure is built. In order to start deploying in a controlled manner and to give enterprise customers a prefix, the TSP framework is used. The TSP server can be put in the core, in the aggregation points or in the pops to offer the service to the customers. IPv6 over IPv4 encapsulation can be used. If the customers are behind an IPv4 NAT, then IPv6 over UDP-IPv4 encapsulation can be used. TSP can be used in combination of other techniques.

5.2 Applicability of TSP in Provider Networks with Home/Small Office Customers

In a provider network where IPv4 is dominant, a tunnelled infrastructure can be used to provider IPv6 services to the home/ small office customers, before a full IPv6 native infrastructure is built. The small networks such as Home/Small offices have a non-upgradable gateway with NAT. TSP with NAT traversal is used to offer IPv6 connectivity and a prefix to the internal network.

Automation of the prefix assignment and DNS delegation, done by TSP, is a very important feature for a provider in order to substantially decrease support costs. The provider can use the same authentication database that is used to authenticate the IPv4 users. Customers can deploy home IPv6 networks without any intervention of the provider support people.

With the NAT discovery function of TSP, providers can use the same TSP infrastructure for both NAT and not-NAT parts of the network.

5.3 Applicability of TSP in Enterprise Networks

In an enterprise network where IPv4 is dominant, a tunnelled infrastructure can be used to provider IPv6 services to the IPv6 islands (hosts or networks) inside the enterprise, before a full IPv6 native infrastructure is built. TSP can be used to give IPv6 connectivity, prefix and routing for the islands. This gives to the enterprise a full control deployment of IPv6 while maintaining automation and permanence of the IPv6 assignments to the islands.

5.4 Applicability of TSP in Wireless Networks

In a wireless network where IPv4 is dominant, hosts and networks move and change IPv4 address. TSP enables the automatic re-establishment of the tunnel when the IPv4 address change.

In a wireless network where IPv6 is dominant, hosts and networks move. TSP enables the automatic re-establishment of the tunnel together with the DSTM mechasnism.

5.5 Applicability of TSP in Unmanaged networks

An unmanaged network is where no network manager or staff is available to configure network devices. TSP is particularly powerful in this context where automation of all necessary information for the IPv6 connectivity is handled by TSP: tunnel end-points parameters,

prefix assignment, dns delegation, routing.

An unmanaged network may be behind a NAT, maybe not. With the NAT discovery function, TSP works automatically in both cases.

5.6 Applicability of TSP for Mobile Hosts

Mobile hosts are common and used. Laptops moving from wireless, wired in office, home, ... are examples. They often have IPv4 connectivity, but not necessarily IPv6. TSP framework enables the mobile hosts to have IPv6 connectivity wherever they are, by having the TSP client sends updated information of the new environment to the TSP server, when a change occur. Together with NAT discovery, the mobile host can be always IPv6 connected wherever it is.

Mobile here means only the change of IPv4 address. MobileIP mechanisms and fast handoff take care of additional constraints in mobile environments.

5.7 Applicability of TSP for Mobile Networks

Mobile networks share the applicability of the mobile hosts. Moreover, in the TSP framework, they also keep their prefix assignment and can control the routing. NAT discovery can also be used.

6. IANA Considerations

A tunnel type registry should be setup by IANA. The following strings are defined in this document: "v6v4" for IPv6 in IPv4 encapsulation (using IPv4 protocol 41) "v6udpv4" for IPv6 in UDP in IPv4 encapsulation "v6anyv4" for IPv6 in IPv4 or IPv6 in UDP in IPv4 encapsulation "v4v6" for IPv4 in IPv6 encapsulation.

Details on the registration procedure for new tokens TBD.

IANA assigned 3653 as the TSP port number.

7. Security Considerations

Authentication of the TSP session uses the SASL[RFC2222] framework,

where the authentication mechanism is negociated between the client and the server. The framework enables to use the level of authentication needed for securing the session, based on the policies.

Static tunnels are created when the TSP negociation is terminated. Static tunnels are not open gateways and exhibit less security issues than automated tunnels. Static IPv6 in IPv4 tunnels security considerations are described in [RFC2893].

8. Conclusion

The Tunnel Setup Protocol (TSP) is applicable in many environments, such as: providers, enterprises, wireless, unmanaged networks, mobile hosts and networks. TSP gives the two tunnel end-points the ability tonegociate tunnel parameters, as well as prefix assignment, dns delegation and routing in an authenticated session. It also provides IPv4 NAT discovery function by using the most effective encapsulation. It also supports the IPv4 mobility of the nodes.

9. Acknowledgements

This draft is the merge of many previous drafts about TSP. Authors who have contributed to earlier drafts are: Florent Parent and Octavio Medina.

References

[1] Myers, J., "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997.

[2] Gilligan, R. and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers", RFC 2893, August 2000.

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

[4] Bound, J., "Dual Stack Transition Mechanism (DSTM)", draft-ietf-ngtrans-dstm-08 (work in progress), July 2002.

[5] Hagino, J., "Possible abuse against IPv6 transition technologies", July 2000.

Author's Address

Marc Blanchet
Hexago
2875 boul. Laurier, suite 300
Sainte-Foy, QC G1V 2M2
Canada

Phone: +1 418 266 5533
EMail: Marc.Blanchet@hexago.com
URI: http://www.hexago.com/

Appendix A. DTD

```
<?xml version="1.0"?>
<!DOCTYPE tunnel  [
<!ELEMENT tunnel        (server?,client?,broker?)>
  <!ATTLIST tunnel action   (create|info|list) #REQUIRED >
  <!ATTLIST tunnel type     (v6v4|v4v6|v6anyv4|v6udpv4|broker) #REQUIRED >
  <!ATTLIST tunnel lifetime CDATA              "1440"    >
<!ELEMENT server        (address+,router?)>

<!ELEMENT client        (address+,router?)>

<!ELEMENT broker        (adress+)>

<!ELEMENT router        (prefix?,dns_server?,as?)>

<!ELEMENT dns_server    (address+)>

<!ELEMENT prefix        (#PCDATA)>
  <!ATTLIST prefix length CDATA #REQUIRED>

<!ELEMENT address       (#PCDATA)>
  <!ATTLIST address type (ipv4|ipv6|dn) #REQUIRED>
]>
```

Appendix B. Error codes

Error codes are sent as a numeric value followed by a text message describing the code. The Tunnel Setup Protocol defines error code numbers 1 through 499 and 1000 through 1499. Profile dependant error codes are defined within the 500 through 999 and 1500 through 1999 range.

The predefined values are :

if a list of tunnel servers is following the error code as a referal service, then 1000 is added to the error code.

200 Success

Successful operation

300 Authentication failed

Invalid userid, password or authentication mechanism.

301 No more tunnels available

The server has reached its capacity limit.

302 Unsupported client version

The client version is not supported by the server.

303 Unsupported tunnel type

The server does not provide the requested tunnel type.

306 Address Pool Exhausted

307 Configuration Error at TEP

308 Requested Address Unavailable

309 Invalid IPv6 address

501 Invalid IPv4 address

502 Invalid or duplicate nicname

504 Router function not supported

506 IPv4 prefix already used for existing tunnel

507 Requested prefix length cannot be assigned

509 DNS delegation setup error

514 Protocol error

517 Unsupported router protocol

518 Unsupported prefix length

520 Missing prefix length

Full Copyright Statement

Acknowledgement

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