Network Working Group
Internet-Draft
Expires: August 15, 2004

F. Templin
Nokia
T. Gleeson
Cisco Systems K.K.
M. Talwar
D. Thaler
Microsoft Corporation
February 16, 2004

Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)
draft-ietf-ngtrans-isatap-20.txt

Status of this Memo

Copyright Notice

Abstract

The Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) connects IPv6 hosts/routers over IPv4 networks. ISATAP views the IPv4 network as a link layer for IPv6 and views other nodes on the network as potential IPv6 hosts/routers. ISATAP supports automatic tunneling and a tunnel interface management abstraction similar to the Non-Broadcast, Multiple Access (NBMA) and ATM Permanent/Switched Virtual Circuit (PVC/SVC) models.

Table of Contents

1. Introduction

This document specifies a simple mechanism called the Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) that connects IPv6 [RFC2460] hosts/routers over IPv4 [STD5] networks. Dual-stack (IPv6/IPv4) nodes use ISATAP to automatically tunnel IPv6 packets in IPv4, i.e., ISATAP views the IPv4 network as a link layer for IPv6 and views other nodes on the network as potential IPv6 hosts/routers.

ISATAP enables automatic tunneling whether global or private IPv4 addresses are used, and supports a tunnel interface management abstraction similar to the Non-Broadcast, Multiple Access (NBMA) [RFC2491] and ATM Permanent/Switched Virtual Circuit (PVC/SVC) [RFC2492] models.

The main objectives of this document are to: 1) describe the ISATAP conceptual model, 2) specify addressing requirements, 3) discuss configuration and management requirements, 4) specify automatic tunneling using ISATAP, 5) specify operational aspects of IPv6 Neighbor Discovery, and 6) discuss IANA and Security considerations.

This document surveys all IETF v6ops WG documents current up to February 16, 2004.

2. Requirements

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [BCP14].

This document also makes use of internal conceptual variables to describe protocol behavior and external variables that an implementation must allow system administrators to change. The specific variable names, how their values change, and how their settings influence protocol behavior are provided to demonstrate protocol behavior. An implementation is not required to have them in

the exact form described here, so long as its external behavior is consistent with that described in this document.

3. Terminology

The terminology of [STD3][RFC2460][RFC2461][RFC3582] applies to this document. The following additional terms are defined:

ISATAP node:
a node that implements the specifications in this document.

ISATAP daemon:
an ISATAP node's server application that uses an API for control plane signaling and tunnel interface configuration/management.

ISATAP driver:
an ISATAP node's network module that provides an API for control plane signaling and tunnel interface configuration/management. Also provides a packet encapsulation/decapsulation engine, and an embedded gateway function (see: [STD3], section 3.3.4.2).

logical interface:
an IPv6 address or a configured tunnel interface associated with an ISATAP interface (see: [STD3], section 3.3.4.1).

ISATAP interface:
an ISATAP node's point-to-multipoint interface that provides a control plane interface for the ISATAP daemon and a forwarding plane nexus for its associated logical interfaces.

ISATAP interface identifier:
an IPv6 interface identifier with an embedded IPv4 address constructed as specified in section 6.1.

ISATAP address:
an IPv6 unicast address assigned on an ISATAP interface with an on-link prefix and an ISATAP interface identifier.

locator:
an IPv4 address-to-interface mapping, i.e., a node's IPv4 address

and the index for it's associated interface.

locator set:
a set of locators associated with a tunnel interface, where each locator in the set belongs to the same site.

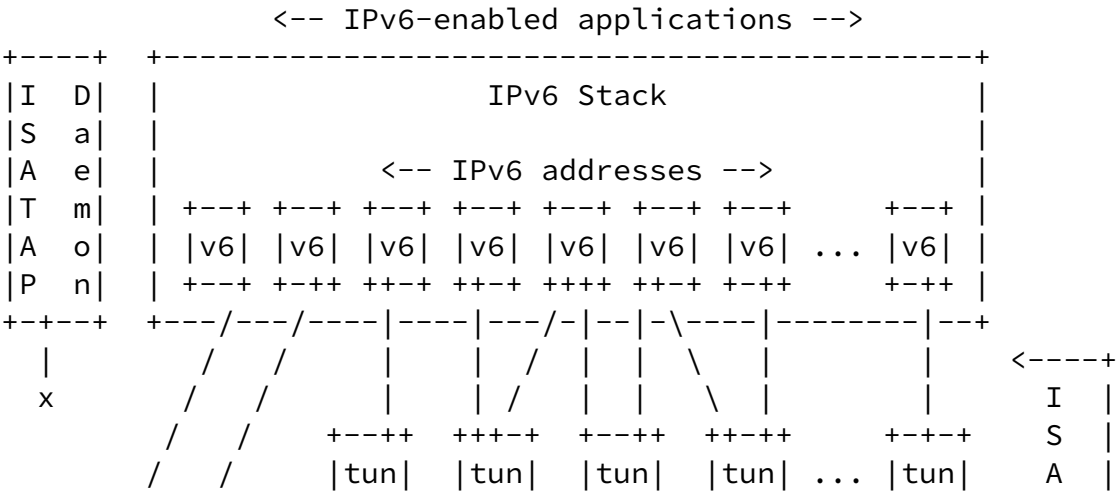
4. ISATAP Conceptual Model

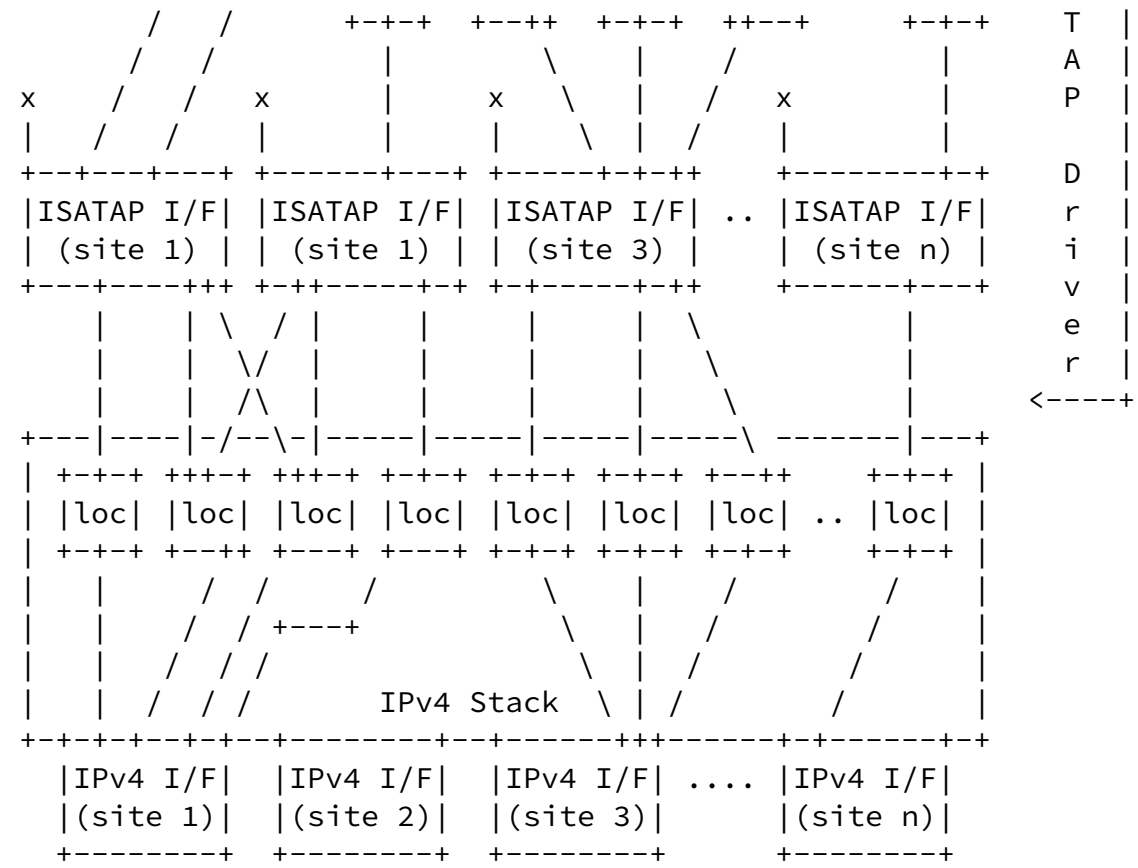
ISATAP interfaces are advertising IPv6 interfaces that provide a point-to-multipoint abstraction for IPv6-in-IPv4 tunneling. They provide a forwarding plane nexus (used by the ISATAP driver) for their associated logical interfaces. They also provide a control plane interface (used by the ISATAP daemon) for tunnel configuration signaling.

The ISATAP driver encapsulates packets for transmission according to parameters associated with its logical interfaces. It also determines the correct interface to receive each tunneled packet after decapsulation, and provides an embedded gateway function.

The ISATAP daemon configures and manages tunnels via an API provided by the ISATAP driver. Each such configured tunnel provides a nexus for multiple applications using IPv6 addresses as application identifiers. Each such application identifier provides a nexus for multiple sessions. In summary, each configured tunnel provides a point-to-point connection between peers that can support multiple applications and multiple instances of each application.

The following example diagram depicts the ISATAP conceptual model:





5. Node Requirements

ISATAP nodes observe the common functionality requirements in [NODEREQ] and the DNS requirements in ([MECH], section 2.2). They also implement the additional features specified in this document.

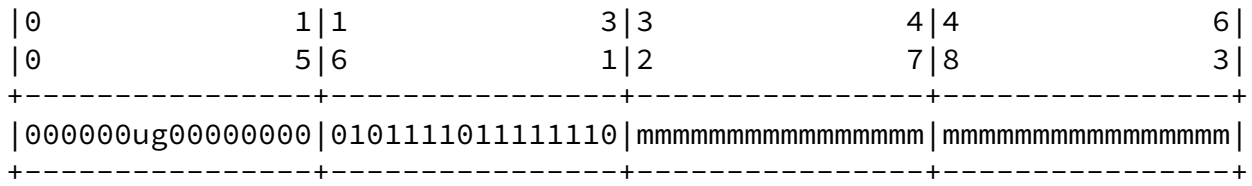
6. Addressing Requirements

ISATAP nodes implement the addressing requirements found in ([NODEREQ], section 4.5). [RFC2462][RFC3484][ADDR] MUST be supported, and [RFC3041] SHOULD be supported (configurable on a per-connection basis).

6.1 ISATAP Interface Identifiers

ISATAP interface identifiers are constructed in Modified EUI-64 format ([ADDR], appendix A). They are formed by concatenating the 24-bit IANA OUI (00-00-5E), the 8-bit hexadecimal value 0xFE, and a 32-bit IPv4 address in network byte order.

The format for ISATAP interface identifiers is given below (where 'u' is the IEEE universal/local bit, 'g' is the IEEE group/individual bit, and the 'm' bits represent the concatenated IPv4 address):



When the IPv4 address is known to be globally unique, the 'u' bit is set to 1; otherwise, the 'u' bit is set to 0 ([ADDR], section 2.5.1). See: Appendix C for additional non-normative details.

6.2 ISATAP Addresses

Any IPv6 unicast address ([ADDR], section 2.5) that contains an ISATAP interface identifier constructed as specified in section 6.1 and an on-link prefix on an ISATAP interface is considered an ISATAP address.

6.3 Multicast/Anycast

ISATAP interfaces recognize a node's required IPv6 multicast/anycast addresses ([ADDR], section 2.8).

For IPv6 multicast addresses of interest to local applications, ISATAP nodes join the corresponding Organization-Local Scope IPv4 multicast groups ([RFC2529], section 6) on each interface that appears in an ISATAP interface's locator set (see: section 7.2).

IPv6 multicast addresses of interest include a node's required multicast addresses, and may also include e.g, the 'All_DHCP_Relay_Agents_and_Servers' and 'All_DHCP_Servers' multicast

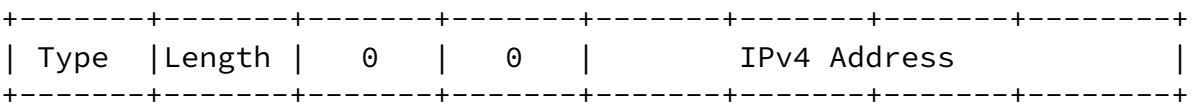
addresses (i.e., if the node is configured as a DHCPv6 server

[RFC3315][RFC3633]), etc.

Considerations for IPv6 anycast appear in [ANYCAST].

6.4 Source/Target Link Layer Address Options

Source/Target Link Layer Address Options ([RFC2461], section 4.6.1) for ISATAP have the following format:



Type:
1 for Source Link-layer address. 2 for Target Link-layer address.

Length:
1 (in units of 8 octets).

IPv4 Address:
A 32 bit IPv4 address, in network byte order.

ISATAP nodes use the specifications in ([MECH], section 3.8) that pertain to sending and receiving Source/Target Link Layer Address Options.

7. Configuration and Management Requirements

7.1 Network Management

This document defines no new MIB tables, nor extensions to any existing MIB tables. Objects found in [FTMIB][IPMIB][TUNMIB] are supported as described in the following subsections.

7.2 The ifRcvAddressTable

The ISATAP driver maintains ifRcvAddressTable as a bidirectional association of locators with tunnel interfaces. Each locator in the table includes an IPv4 address-to-interface mapping (i.e., an IPv4 ipAddressEntry in the node's ipAddressTable) and a list of associated tunnel interfaces. Each tunnel interface in the table has a tunnelIfEntry and a list of associated locators, i.e., a "locator set".

The ISATAP driver implements the following conceptual functions to manage and search the ifRcvAddressTable:

7.2.1 RcvTableAdd(locator, tunnel_interface)

Creates a bidirectional association in the ifRcvAddressTable between the locator and tunnel interface, i.e., adds the locator to the tunnel interface's locator set and adds the tunnel interface to the locator's association list.

Returns success or failure.

7.2.2 RcvTableDel(locator, tunnel_interface)

Deletes ifRcvAddressTable entries according to the locator and tunnel interface arguments as follows:

- if both arguments are NULL, garbage-collects the entire table.

- if both arguments are non-NULL, deletes the locator from the tunnel interface's locator set and deletes the tunnel interface from the locator's association list.

- if the locator is non-NULL and tunnel interface is NULL, deletes the locator from the locator sets of all tunnel interfaces.

- if the locator is NULL and the tunnel interface is non-NULL, deletes the tunnel interface from the association lists of all locators.

Returns success or failure.

7.2.3 RcvTableLocate(packet)

Searches the ifRcvAddressTable to locate the correct tunnel interface to decapsulate a packet. First, determines the locator that matches the packet's IPv4 destination address and ifIndex for the interface the packet arrived on. Next, checks each tunnel interface in the locator's association list for exact matches of tunnelIfEncapsMethod with the packet's encapsulation type and tunnelIfRemoteInetAddress with the packet's IPv4 source address.

If there is no match on the packet's IPv4 source address, a tunnel interface with a matching tunnelIfEncapsMethod and with tunnelIfRemoteInetAddress set to 0.0.0.0 is selected. If there are multiple matches, a tunnel interface with tunnelIfLocalInetAddress that matches the packet's IPv4 destination address is preferred.

Returns a pointer to a tunnel interface if a match is found; else

NULL.

7.3 ISATAP Driver API

The ISATAP driver implements an API used by, e.g., the ISATAP daemon, startup scripts, manual command line entry, kernel processes, etc. Access MUST be restricted to privileged users and applications. ISATAP nodes implement the basic and advanced APIs for IPv6 [RFC3493][RFC3542].

7.4 ISATAP Interface Creation/Configuration

ISATAP interfaces are created via the tunnelIfConfigTable, which results in simultaneous creation of a tunnelIfEntry and a companion ipv6InterfaceEntry. Each ISATAP interface configures a locator set, where each locator in the set represents an IPv4 address-to-interface mapping for the same site (or, represents a mapping that is routable on the global Internet). ISATAP interfaces MUST NOT configure a locator set that spans multiple sites.

ISATAP interfaces configure the following values for objects in tunnelIfEntry:

- tunnelIfEncapsMethod is set to an IANATunnelType for "isatap".

- tunnelIfLocalInetAddress is set to an IPv4 address from the interface's locator set.

- tunnelIfRemoteInetAddress is set to 0.0.0.0 to denote wildcard match for remote tunnel endpoints.

- other read-write objects in the tunnelIfEntry are configured as for any tunnel interface.

ISATAP interfaces are configured as advertising IPv6 interfaces and set the following values for objects in ipv6InterfaceEntry:

- ipv6InterfaceType is set to "tunnel".

- ipv6InterfacePhysicalAddress is set to an octet string of zero length to indicate that this IPv6 interface does not have a

physical address.

- ipv6InterfaceForwarding and ip6Forwarding for the node are set to "forwarding".

- other read-write objects in ipv6InterfaceEntry are configured as for any IPv6 interface.

ISATAP interfaces create an ipv6RouterAdvertEntry and set its

ipv6RouterAdvertIfIndex object to the same value as ipv6InterfaceIfIndex. Other objects in ipv6RouterAdvertEntry are configured as for any IPv6 router.

7.5 Configured Tunnel Creation/Configuration

Configured tunnels are normally created by the ISATAP daemon in dynamic response to a tunnel creation request as an ISATAP interface's associated logical interface; they inherit the locator set of their associated ISATAP interface. Configured tunnels set the following values for objects in tunnelIfEntry:

- tunnelIfEncapsMethod is set to an appropriate IANATunnelType value.

- tunnelIfLocalInetAddress is set to an IPv4 address from the interface's locator set.

- tunnelIfRemoteInetAddress is set to an IPv4 address for the node at the far end of the tunnel.

- other read-write objects in the tunnelIfEntry are configured as for any tunnel interface.

Configured tunnels set values for objects in ipv6InterfaceEntry as follows:

- ipv6InterfaceType is set to "tunnel".

- ipv6InterfacePhysicalAddress is set to an octet string of zero length to indicate that this IPv6 interface does not have a physical address.

- other read-write objects in ipv6InterfaceEntry are configured as for any IPv6 interface.

7.6 Reconfigurations Due to IPv4 Address Changes

When an IPv4 address is removed from an interface, its corresponding locator SHOULD be removed from all locator sets via RcvTableDel(locator, NULL); tunnelIfEntrys that used the IPv4 address as tunnelIfLocalInetAddress SHOULD also configure a different local IPv4 address from their remaining locator set.

When a new IPv4 address is added to an IPv4 interface, the node MAY add the corresponding new locator to a tunnel interface's locator set via RcvTableAdd(locator, tunnel_interface), and MAY also set tunnelIfLocalInetAddress for its tunnelIfEntry to the new address.

Methods for triggering the above changes are out of scope.

8. Automatic Tunneling

ISATAP nodes use the basic tunneling mechanisms specified in [MECH]. The following additional specifications are also used:

8.1 Encapsulation

The ISATAP driver encapsulates IPv6 packets using various encapsulation methods, including ip-protocol-41 (e.g., 6over4 [RFC2529], 6to4 [RFC3056], IPv6-in-IPv4 configured tunnels [MECH], isatap, etc.), UDP [STD6] port 3544, and others.

Security processing (e.g., [RFC2402][RFC2406], etc.), upper layer fragmentation [RFC3542] and header compression for the packet's inner headers are performed prior to encapsulation.

8.1.1 NAT Traversal

Native IPv6 and/or ip-protocol-41 encapsulation provides sufficient functionality to support communications between peers that reside within the same site (i.e., the same enterprise network). When the remote peer is in a different site, NAT traversal via UDP/IPv4 encapsulation MAY be necessary.

When an ISATAP node determines that NAT traversal is necessary to reach a particular peer, it encapsulates IPv6 packets using UDP/IPv4 port 3544 encapsulation. This determination may come through, e.g., first attempting communications via ip-protocol-41 then failing over to UDP/IPv4 port 3544 encapsulation, administrative knowledge that a NAT is on the path, etc.

8.1.2 Multicast

ISATAP interfaces encapsulate packets with IPv6 multicast destination addresses using a mapped Organization-Local Scope IPv4 multicast address ([RFC2529], section 6) as the destination address in the encapsulating IPv4 header.

8.2 Tunnel MTU and Fragmentation

Encapsulated packets sent by the ISATAP driver may require host-based IPv4 fragmentation in order to satisfy the 1280 byte IPv6 minimum MTU, e.g., when the underlying link has a small IPv4 MTU [BCP48]. While this intentional fragmentation is not considered harmful, unmitigated IPv4 fragmentation caused by the network can cause poor performance [FRAG]. For example, since the minimum IPv4 fragment size is only 8 bytes [STD5], a single 1280 byte encapsulated packet could be shredded by the network into as many as 160 IPv4 fragments with obvious negative performance implications.

ISATAP uses the MTU and fragmentation specifications in ([MECH], section 3.2) and the Maximum Reassembly Unit (MRU) specifications in ([MECH], section 3.6), which provide sufficient measures for avoiding excessive IPv4 fragmentation in certain controlled environments (e.g., 3GPP operator networks, enterprise networks, etc). To minimize IPv4 fragmentation and improve performance in general use case

scenarios, ISATAP nodes SHOULD add the following simple instrumentation to the IPv4 reassembly cache:

When the initial fragment of an encapsulated packet arrives, the packet's IPv4 reassembly timer is set to 1 second (i.e., the worst case store-and-forward delay budget for a 1280 byte packet). If an encapsulated packet's IPv4 reassembly timer expires:

- If enough contiguous leading bytes of the packet have arrived (see: section 8.6), reassemble the packet using zero-filled or heuristically-chosen replacement data bytes in place of any missing fragments. (Otherwise, garbage-collect the reassembly buffer and return from processing.)

- Mark the packet as "INCOMPLETE", and also mark it with an "ACTUAL_BYTES" length that encodes the actual number of data bytes in fragments that arrived.

- Deliver the packet to the ISATAP driver, and do not send an ICMPv4 "time exceeded" message [STD5].

Appendix B provides informative text on the derivation of the 1280 byte IPv6 minimum MTU.

8.3 Handling ICMPv4 Errors

ISATAP interfaces SHOULD process ARP failures and persistent ICMPv4 errors as link-specific information indicating that a path to a neighbor may have failed ([RFC2461], section 7.3.3).

8.4 Link-Local Addresses

ISATAP interfaces use link local addresses constructed as specified in section 6.1 of this document.

8.5 Neighbor Discovery over Tunnels

The specification in ([MECH], section 3.8) is used; the additional specification for neighbor discovery in section 9 of this document are also used.

8.6 Decapsulation/Filtering

ISATAP nodes typically arrange for the ISATAP driver to receive all IPv4-encapsulated IPv6 packets that are addressed to one of the node's IPv4 addresses. Examples include ip-protocol-41 (e.g., 6to4, 6over4, configured tunnels, isatap, etc.), UDP/IPv4 port 3544, and others. The ISATAP driver uses the decapsulation and filtering specifications in ([MECH], section 3.6), and processes each packet according to the following steps:

1. Locate the correct tunnel interface to receive the packet (see: section 7.2.3). If not found, silently discard the packet and return from processing.

2. If the tunnel uses header compression, reconstitute headers. If header reconstitution fails, silently discard the packet and return from processing.

3. Verify that the packet's IPv4 source address is correct for the encapsulated IPv6 source address. For packets received on a configured tunnel interface, verification is exactly as specified in ([MECH], section 3.6).

   For packets received on an ISATAP interface, the IPv4 source address is correct if:

   - the IPv6 source address is an ISATAP address that embeds the IPv4 source address in its interface identifier, or:

   - the IPv6 source address is the address of an IPv6 neighbor on an ISATAP interface associated with the locator that matched the packet (see: section 7.2.3), or:

   - the IPv4 source address is a member of the Potential Router List (see: section 9.1).

   If the IPv4 source address is incorrect, silently discard the

   packet and return from processing.

4. Perform IPv4 ingress filtering (optional; disabled by default) then decapsulate the packet but do not discard encapsulating

headers. If the IPv6 source address is invalid (see: [MECH], section 3.6), silently discard the packet and return from processing.

For UDP port 3544 packets received on an ISATAP interface, if the IPv6 source address is an ISATAP link local address with the 'u' bit set to 0 and an embedded IPv4 address that does not match the IPv4 source address, rewrite the IPv6 source address to inform upper layers of the sender's mapped UDP port number and IPv4 source address. Specific rules for rewriting the IPv6 source address are established during ISATAP interface configuration.

5. Perform ingress filtering on the IPv6 source address (see: [MECH], section 3.6). Next, determine the correct transport protocol listener [FLOW] if the packet is destined to the localhost; otherwise, perform an IPv6 forwarding table lookup and site border/firewall filtering (see: [UNIQUE], section 6).

   If the packet cannot be delivered, the driver SHOULD send an ICMPv6 Destination Unreachable message ([RFC2463], section 3.2) to the packet's source. The message SHOULD select as its source address an IPv6 address from the outgoing interface (if the packet was destined to the localhost) or an ingress-wise correct IPv6 address from the interface that would have forwarded the packet had it not been filtered.

   The Code field of the message is set as follows:

   - if there is no route to the destination, the Code field is set to 0 (see: [RFC2463], section 3.1).

   - if communication with the destination is administratively prohibited, the Code field is set to 1 ([RFC2463], section 3.1).

   - if the packet is destined to the localhost, but the transport protocol has no listener, the Code field is set to 4 ([RFC2463], section 3.1).

   - if the packet's destination is beyond the scope of the source address, the Code field is set to 2 (see: IANA Considerations).

   - if the packet was dropped due to ingress filtering policies,

the Code field is set to 5 (see: IANA Considerations).

- if the packet is dropped due to a reject route, the Code field is set to 6 (see: IANA Considerations).

- if the packet was received on a point-to-point link and destined to an address within a subnet assigned to that same link, or if the reason for the failure to deliver cannot be mapped to any of the specific conditions listed above, the Code field is set to 3 ([RFC2463], section 3.2).

After sending the ICMPv6 Destination Unreachable message, discard the packet and return from processing.

6. If the packet is "INCOMPLETE" (see section 8.2) prepare an authenticated, unsolicited Router Advertisement message ([RFC2461], section 6.2.4) with an MTU option that encodes the maximum of "ACTUAL_BYTES" and (68 bytes minus the size of encapsulating headers.)

   The IPv6 destination address in the Router Advertisement message is set to the packet's IPv6 source address, and the message is reverse-encapsulated and returned to the node that sent the "INCOMPLETE" packet, i.e., it is NOT presented to the native IPv6 stack for transmission.

   The 68 byte minimum MTU is due to the requirement that every Internet module must be able to forward a datagram of 68 octets without further fragmentation ([STD5], Internet Protocol, section 3.2).

7. Discard encapsulating headers. If the packet was destined to a remote host, forward the packet and return from processing. Otherwise, apply security processing (e.g., [RFC2402][RFC2406], etc.), and place the packet in a buffer for upper layers. The buffer may be, e.g., the IPv6 reassembly cache, an application's mapped data buffer [RFC3542], etc.

   If there is clear evidence that upper layer reassembly has stalled, an ICMPv6 Packet Too Big message [RFC1981] MAY be sent to the packet's source address with an MTU value likely to incur successful reassembly. Some applications may realize greater efficiency by accepting partial information from "INCOMPLETE" packets (see: section 8.2) and requesting selective retransmission of missing portions.

9. Neighbor Discovery for ISATAP Interfaces

ISATAP nodes use the neighbor discovery mechanisms specified in [RFC2461] to create/change neighbor cache entries and to provide control plane signaling for automatic tunnel configuration. Securing mechanisms for neighbor discovery messages (e.g., [RFC2402], [SEND]) are used according to the trust model appropriate for the given deployment scenario [SENDPS]. ISATAP interfaces also implement the following specifications:

9.1 Conceptual Model Of A Host

To the list of Conceptual Data Structures ([RFC2461], section 5.1), ISATAP interfaces add:

Potential Router List
: A set of entries about potential routers; used to support the mechanisms specified in section 9.2.2.1. Each entry ("PRL(i)") has an associated timer ("TIMER(i)"), and an IPv4 address ("V4ADDR(i)") that represents a router's advertising ISATAP interface.

9.2 Router and Prefix Discovery

9.2.1 Router Specification

Solicited Router Advertisements sent on ISATAP interfaces are sent directly to the soliciting node, i.e., they might not be received by other nodes on the link.

Router Advertisements sent on ISATAP interfaces MAY include information delegated via DHCPv6 [RFC3633].

Router Advertisements sent on ISATAP interfaces MUST NOT include a prefix option containing a preferred lifetime greater than the valid lifetime.

9.2.2 Host Specification

The Host Specification in ([RFC2461], section 6.3) is used. ISATAP

interfaces add the following specifications:

9.2.2.1 Host Variables

To the list of host variables ([RFC2461], section 6.3.2), ISATAP interfaces add:

PrlRefreshInterval
: Time in seconds between successive refreshments of the PRL after initialization. The designated value of all 1's (0xffffffff) represents infinity.
Default: 3600 seconds

MinRouterSolicitInterval
: Minimum time in seconds between successive solicitations of the same advertising ISATAP interface. The designated value of all 1's (0xffffffff) represents infinity.

9.2.2.2 Potential Router List Initialization

ISATAP nodes provision an ISATAP interface's PRL with IPv4 addresses discovered via a DNS fully-qualified domain name (FQDN) [STD13], manual configuration, a DHCPv4 option, a DHCPv4 vendor-specific option, or an unspecified alternate method.

FQDNs are established via manual configuration or an unspecified alternate method. FQDNs are resolved into IPv4 addresses through querying the DNS service, querying a site-specific name service, static host file lookup, or an unspecified alternate method.

When the node provisions an ISATAP interface's PRL with IPv4 addresses, it sets a timer for the interface (e.g., PrlRefreshIntervalTimer) to PrlRefreshInterval seconds. The node re-initializes the PRL as specified above when PrlRefreshIntervalTimer expires, or when an asynchronous re-initialization event occurs. When the node re-initializes the PRL, it resets PrlRefreshIntervalTimer to PrlRefreshInterval seconds.

9.2.2.3 Processing Received Router Advertisements

To the list of checks for validating Router Advertisement messages ([RFC2461], section 6.1.1), ISATAP interfaces add:

- IP Source Address is an ISATAP link-local address that embeds V4ADDR(i) for some PRL(i).

Valid Router Advertisements received on an ISATAP interface are processed exactly as specified in ([RFC2461], section 6.3.4). [RFC3315] and [RFC3633] are stateful mechanisms associated with the M and O bits.

For Router Advertisements that include an MTU option, the MTU value does not alter the ISATAP interface LinkMTU. Instead, the MTU value is recorded, e.g., in the IPv6 forwarding table. If the IPv6 destination address is one of the node's own unicast addresses, the

MTU value is recorded such that upper layer fragmentation [RFC3542] will be used to reduce the size of the encapsulated packets sent via the router. The recorded value is aged as for IPv6 path MTU information [RFC1981].

9.2.2.4 Sending Router Solicitations

To the list of events after which Router Solicitation messages may be sent ([RFC2461], section 6.3.7), ISATAP interfaces add:

- TIMER(i) for some PRL(i) expires.

Since unsolicited Router Advertisements may be incomplete and/or absent, TIMER(i) MAY be used to schedule periodic Router Solicitation events for certain PRL(i)'s.

When used, TIMER(i) SHOULD be set such that the next Router Solicitation event will refresh remaining lifetimes stored for the PRL(i), including Router Lifetime, Valid Lifetimes received in Prefix Information Options, and Route Lifetimes received in Route Information Options [DEFLT]. TIMER(i) MUST be set to no less than MinRouterSolicitInterval seconds, where MinRouterSolicitInterval is configurable for the node (or, for each PRL(i)) with a conservative default value.

When TIMER(i) expires, Router Solicitation messages are sent as

specified in ([RFC2461], section 6.3.7) except that the messages are sent directly to PRL(i), i.e., they might not be received by other nodes on the link. While the node continues to use PRL(i) as a router (and, while PRL(i) continues to act as a router), the node resets TIMER(i) after each expiration as described above.

9.3 Address Resolution and Neighbor Unreachability Detection

9.3.1 Address Resolution

The specification in ([RFC2461], section 7.2) is used. ISATAP addresses for which the neighbor's link-layer address cannot otherwise be determined (e.g., from a neighbor cache entry) are resolved to link-layer addresses by a static computation, i.e., the last four octets are treated as an IPv4 address.

Hosts SHOULD perform an initial reachability confirmation by sending Neighbor Solicitation message(s) and receiving a Neighbor Advertisement message. Routers MAY perform this initial reachability confirmation, but this might not scale in all environments.

9.3.2 Neighbor Unreachability Detection

Hosts SHOULD perform Neighbor Unreachability Detection ([RFC2461], section 7.3). Routers MAY perform neighbor unreachability detection, but this might not scale in all environments.

10. Security considerations

The Security Considerations in the normative references, and in ([NODEREQ], section 8) apply. Also:

- ISATAP nodes observe the security considerations outlined in [SENDPS]; use of IPsec (e.g., [RFC2402][RFC2406], etc.) is not always feasible.

- site border routers SHOULD install a reject route for the IPv6 prefix FC00::/7 [UNIQUE] to insure that packets with local IPv6 destination addresses will not be forwarded outside of the site via a default route.

- administrators MUST ensure that lists of IPv4 addresses representing the advertising ISATAP interfaces of PRL members are well maintained.

11. IANA Considerations

The IANA is instructed to specify the format for Modified EUI-64 address construction ([ADDR], Appendix A) in the IANA Ethernet Address Block. The text in Appendix C of this document is offered as an example specification. The current version of the IANA registry for Ether Types can be accessed at:
http://www.iana.org/assignments/ethernet-numbers.

The IANA is instructed to assign the new ICMPv6 code field types found in Appendix D of this document for the ICMPv6 Destination Unreachable message. The policy for assigning new ICMPv6 code field types is First Come First Served, as defined in [BCP26]. The current version of the IANA registry for ICMPv6 type numbers can be accessed at:
http://www.iana.org/assignments/icmpv6-parameters.

12. IAB Considerations

[RFC3424] ("IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation") section 4 requires that any proposal supporting NAT traversal must explicitly address the following considerations:

12.1 Problem Definition

The specific problem being solved is enabling IPv6 connectivity for ISATAP nodes that are unable to communicate via ip-protocol-41 or native IPv6.

12.2 Exit Strategy

ISATAP nodes use UDP/IPv4 encapsulation for NAT traversal as a last resort. As soon as native IPv6 or ip-protocol-41 support becomes available, ISATAP nodes will naturally cease using UDP/IPv4 encapsulation.

12.3 Brittleness

UDP/IPv4 encapsulation with ISATAP introduces brittleness into the system in several ways: the discovery process assumes a certain classification of devices based on their treatment of UDP; the mappings need to be continuously refreshed, and addressing structure may cause some hosts located beyond a common NAT to be unreachable from each other.

ISATAP assumes a certain classification of devices based on their treatment of UDP. There could be devices that would not fit into one of these molds, and hence would be improperly classified by ISATAP.

The bindings allocated from the NAT need to be continuously refreshed. Since the timeouts for these bindings is very implementation specific, the refresh interval cannot easily be determined. When the binding is not being actively used to receive traffic, but to wait for an incoming message, the binding refresh will needlessly consume network bandwidth.

12.4 Requirements for a Long Term Solution

The devices that implement the IPv4 NAT service should in the future also become IPv6 routers.

13. Acknowledgments

The ideas in this document are not original, and the authors acknowledge the original architects. Portions of this work were sponsored through SRI International internal projects and government

contracts. Government sponsors include Monica Farah-Stapleton and Russell Langan (U.S. Army CECOM ASEO), and Dr. Allen Moshfegh (U.S. Office of Naval Research). SRI International sponsors include Dr. Mike Frankel, J. Peter Marcotullio, Lou Rodriguez, and Dr. Ambatipudi Sastry.

The following are acknowledged for providing peer review input: Jim Bound, Rich Draves, Cyndi Jung, Ambatipudi Sastry, Aaron Schrader, Ole Troan, Vlad Yasevich.

The following are acknowledged for their significant contributions: Alain Durand, Hannu Flinck, Jason Goldschmidt, Nathan Lutchansky, Karen Nielsen, Mohan Parthasarathy, Chirayu Patel, Art Shelest, Pekka Savola, Margaret Wasserman, Brian Zill.

The authors acknowledge the work of Quang Nguyen on "Virtual Ethernet" under the guidance of Dr. Lixia Zhang that proposed very similar ideas to those that appear in this document. This work was first brought to the authors' attention on September 20, 2002.

IAB considerations are the same as for Teredo. The diagram in section 4 was inspired by a similar diagram in RFC 3371.

The following individuals are acknowledged for their helpful insights on path MTU discovery: Jari Arkko, Iljitsch van Beijnum, Jim Bound, Ralph Droms, Alain Durand, Jun-ichiro itojun Hagino, Brian Haberman, Bob Hinden, Christian Huitema, Kevin Lahey, Hakgoo Lee, Matt Mathis, Jeff Mogul, Erik Nordmark, Soohong Daniel Park, Chirayu Patel, Michael Richardson, Pekka Savola, Hesham Soliman, Mark Smith, Dave Thaler, Michael Welzl, Lixia Zhang and the members of the Nokia NRC/ COM Mountain View team.

"...and I'm one step ahead of the shoe shine,
Two steps away from the county line,
Just trying to keep my customers satisfied,
Satisfi-i-ied!" - Paul Simon, 1969

Appendix A. Major Changes

Major changes from earlier versions to version 17:

- new section on configuration/management.
- new appendices on IPv6 minimum MTU; IANA considerations.
- expanded section on MTU and fragmentation.
- expanded sections on encapsulation/decapsulation.
- specified relation to IPv6 Node Requirements.
- introduced distinction between control; forwarding planes.
- specified multicast mappings.
- revised neighbor discovery, address autoconfiguration, IANA considerations and security considerations sections.

Appendix B. The IPv6 minimum MTU

The 1280 byte IPv6 minimum MTU was proposed by Steve Deering and agreed through working group consensus in November 1997 discussions on the IPv6 mailing list. The size was chosen to allow extra room for link layer encapsulations without exceeding the Ethernet MTU of 1500 bytes, i.e., the practical physical cell size of the Internet. The 1280 byte MTU also provides a fixed upper bound for the size of IPv6 packets/fragments with a maximum store-and-forward delay budget of ~1 second assuming worst-case link speeds of ~10Kbps [BCP48], thus providing a convenient value for use in reassembly buffer timer settings. Finally, the 1280 byte MTU allows transport connections (e.g., TCP) to configure a large-enough maximum segment size for improved performance even if the IPv4 interface that will send the tunneled packets uses a smaller MTU.

Appendix C. Modified EUI-64 Addresses in the IANA Ethernet Address Block

Modified EUI-64 addresses ([ADDR], Appendix A) in the IANA Ethernet Address Block are formed as the concatenation of the 24-bit IANA OUI (00-00-5E) with a 40-bit extension identifier. They have the following appearance in memory (bits transmitted right-to-left within octets, octets transmitted left-to-right):

```
0                       23                                      63
|        OUI            |            extension identifier       |
000000ug00000000 01011110xxxxxxxx xxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxx
```

When the first two octets of the extension identifier encode the hexadecimal value 0xFFFE, the remainder of the extension identifier encodes a 24-bit vendor-supplied id as follows:

```
0                       23              39                      63
|        OUI            |     0xFFFE    |   vendor-supplied id  |
000000ug00000000 0101111011111111 11111110xxxxxxxx xxxxxxxxxxxxxxxx
```

When the first octet of the extension identifier encodes the hexadecimal value 0xFE, the remainder of the extension identifier encodes a 32-bit IPv4 address as follows:

```
0                       23      31                              63
|        OUI            |  0xFE |           IPv4 address        |
000000ug00000000 0101111011111110 xxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxx
```

Modified EUI-64 format interface identifiers are formed by inverting the "u" bit, i.e., the "u" bit is set to one (1) to indicate universal scope and it is set to zero (0) to indicate local scope ([ADDR], section 2.5.1).

Appendix D. Proposed ICMPv6 Code Field Types

Three new ICMPv6 Code Field Type definitions are proposed for the ICMPv6 Destination Unreachable message. The first proposes a new definition for a currently-unassigned code type (2) in the ICMPv6 Type Numbers registry; the others propose new definitions for code types (5) and (6). The code type field definition proposals appear below:

Type	Name	Reference
1	Destination Unreachable	[RFC2463]
Code	2 - beyond the scope of source address	
	5 - source address failed ingress policy	
	6 - reject route to destination	

Normative References

[BCP14] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[BCP26] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[STD3] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[STD5] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[STD6] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC1981] McCann, J., Deering, S. and J. Mogul, "Path MTU Discovery for

IP version 6", RFC 1981, August 1996.

[RFC2402] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[RFC2406] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2461] Narten, T., Nordmark, E. and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[RFC2462] Thomson, S., and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[RFC2463] Conta, A., and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 2463, December 1998.

[RFC2529] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529, March 1999.

[RFC3041] Narten, T. and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 3041, January, 2001.

[RFC3424] Daigle, L. and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[RFC3542] Stevens, W., Thomas, M., Nordmark, E. and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, May 2003.

[RFC3582] Abley, J., Black, B. and V. Gill, "Goals for IPv6 Site-Multihoming Architectures", RFC 3582, August 2003.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.

[ADDR] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", draft-ietf-ipv6-addr-arch-v4, Work in Progress, October 2003.

[DEFLT] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", draft-ietf-ipv6-router-selection, Work in Progress, December 2003.

[MECH] Gilligan, R. and E. Nordmark, "Basic Transition Mechanisms for IPv6 Hosts and Routers", draft-ietf-v6ops-mech-v2, Work in Progress, February 2003.

[NODEREQ] Loughney, J., "IPv6 Node Requirements", draft-ietf-ipv6-node-requirements, Work in Progress, October 2003.

[UNIQUE] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", draft-ietf-ipv6-unique-local-addr, Work in Progress, January 2004.

Informative References

[BCP48] Dawkins, S., Montenegro, G., Kojo, M. and V. Magret, "End-to-end Performance Implications of Slow Links", BCP 48, RFC 3150, July 2001.

[STD13] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2491] Armitage, G., Schulter, P., Jork, M. and G. Harter, "IPv6 over Non-Broadcast Multiple Access (NBMA) networks", RFC 2491, January 1999.

[RFC2492] Armitage, G., Schulter, P. and M. Jork, "IPv6 over ATM Networks", RFC 2492, January 1999.

[RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C. and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[ANYCAST] Hagino, J. and K. Ettikan, "An Analysis of IPv6 Anycast", draft-ietf-ipngwg-ipv6-anycast-analysis, Work in Progress, June 2003.

[FLOW] Rajahalme, J., Conta, A., Carpenter, B. and S. Deering, "IPv6 Flow Label Specification", draft-ietf-ipv6-flow-label, Work in Progress, December 2003.

[FRAG] Mogul, J. and C. Kent, "Fragmentation Considered Harmful", In Proc. SIGCOMM '87 Workshop on Frontiers in Computer Communications Technology. August, 1987.

[FTMIB] Haberman, B. and M. Wasserman, "IP Forwarding Table MIB", draft-ietf-ipv6-rfc2096-update, Work in Progress, August 2003.

[IPMIB] Routhier, S., "Management Information Base for the Internet Protocol (IP)", draft-ietf-ipv6-rfc2011-update, Work in Progress, September 2003.

[SEND] Arkko, J., Kempf, J., Sommerfield, B., Zill, B. and P. Nikander, "Secure Neighbor Discovery (SEND)", draft-ietf-send-ndopt,

Work in Progress, October 2003.

[SENDPS] Nikander, P., Kempf, J. and E. Nordmark, "IPv6 Neighbor Discovery Trust Models and Threats", draft-ietf-send-psreq, Work in Progress, October 2003.

[TUNMIB] Thaler, D., "IP Tunnel MIB", draft-ietf-ipv6-inet-tunnel-mib, Work in Progress, January 2004.

Authors' Addresses

Fred L. Templin
Nokia

313 Fairchild Drive
Mountain View, CA 94110
US

Phone: +1 650 625 2331
EMail: ftemplin@iprg.nokia.com

Tim Gleeson
Cisco Systems K.K.
Shinjuku Mitsu Building
2-1-1 Nishishinjuku, Shinjuku-ku
Tokyo 163-0409
Japan

EMail: tgleeson@cisco.com

Mohit Talwar
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
US

Phone: +1 425 705 3131
EMail: mohitt@microsoft.com

Dave Thaler
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
US

Phone: +1 425 703 8835

EMail: dthaler@microsoft.com

Full Copyright Statement

Intellectual Property

Acknowledgment

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