D3.6.1: Cookbook for IPv6 Renumbering in SOHO and Backbone Networks

Tim Chown, Mark Thompson, Alan Ford, Stig Venaas, Christian Schild, Christian Strauf, Thorsten Kuefer, Frédéric Beck, Olivier Festor, Bartek Gajda

Project Number:	**IST-2001-32603**
Project Title:	**6NET**
CEC Deliverable Number:	**32603/UOS/DS/3.6.1/A1**
Contractual Date of Delivery to the CEC:	31st March 2005
Actual Date of Delivery to the CEC:	10th June 2005
Title of Deliverable:	D3.6.1: Cookbook for IPv6 Renumbering in SOHO and Backbone Networks.
Work package contributing to Deliverable:	WP3
Version:	1.0 (10th June 2005)
Type of Deliverable*:	R
Deliverable Security Class**:	PU
Editors:	Tim Chown (University of Southampton)
Reviewers:	Joao Nuno Ferreira (FCCN)
Contributors:	Tim Chown, Mark Thompson, Alan Ford, Stig Venaas (University of Southampton), Christian Schild, Christian Strauf, Thorsten Kuefer (University of Muentser), Frédéric Beck, Olivier Festor (INRIA-LORIA), Bartek Gajda (PSNC)

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other

** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Abstract:

In this text we present the results of a set of experiments that are designed to be a first step in the process of analysing how effective network renumbering procedures may be in the context of IPv6. An IPv6 site will need to get provider assigned (PA) address space from its upstream ISP. Because provider independent (PI) address space is not available for IPv6, a site wishing to change provider will need to renumber from its old network prefix to the new one. We look at the scenarios, issues and enablers for such renumbering, and present results and initial conclusions and recommendations in the context of SOHO and backbone networking. A subsequent deliverable (D3.6.2) will refine these findings, adding additional results and context from enterprise and ISP renumbering scenarios.

Keywords:

IPv6 renumbering, provider assigned address space, provider independent address space

Table of Contents

Table of Figures

1. Introduction

In this text we present the results of a set of experiments that are designed to be a first step in the process of analysing how effective network renumbering procedures may be in the context of IPv6.

This work was carried out jointly with funding through both 6NET and Cisco.

An IPv6 site will need to get provider assigned (PA) address space from its upstream ISP. Because provider independent (PI) address space is not available for IPv6, a site wishing to change provider will need to renumber from its old network prefix to the new one. This is a particularly important issue for IPv6 deployment, if adopters wish to minimise their dependency on their upstream ISP. The lack of PI address space also impacts IPv6 multihoming, for similar reasons.

We look at the scenarios, issues and enablers for such renumbering, and present results and initial conclusions and recommendations in the context of SOHO and backbone networking.

A subsequent deliverable (D3.6.2) will refine these findings, adding additional results and context from enterprise and ISP renumbering scenarios.

We begin in Section 2 by looking at prior art and work done in the area of IP network renumbering, and the tools available for IPv6 that may simplify the process (compared with IPv4). In Section 3 we review triggers and scenarios for IPv6 renumbering, requirements for renumbering, and considerations from an administrative and network topology point of view, but also from the application perspective (and thus 'survivability' issues in a renumbering event).

Sections 4 and 5 report the initial findings of SOHO oriented renumbering experiments at Southampton and JOIN, where a 'simple' network with one access router and LAN undergoes a renumbering procedure as defined by [BAKER] (which, to simplify, sees the link transition three stable states of original prefix in use, both prefixes in use, and finally only the new prefix in use). Part of the [BAKER] process requires an IPv6 link to operate with two prefixes on a link. In Section 6 we study the issues with such a mode of operation. Such usage may also be interesting from a network multihoming perspective. We will cover the two prefix topic for enterprise networks in D3.6.2).

Section 7 reports JOIN's process of renumbering the 6WiN national backbone network, again following the [BAKER] procedure as far as it applies to that scenario. In Sections 8 and 9 we describe experiments performed by PSNC and Loria/Inria to determine issues with IPv6 network management and monitoring tools where renumbering events are applied, and we report on problems observed.

Section 10 draws on observations in theory and practice to make a set of recommendations for network administrators and designers, application developers, and OS and stack developers, that may assist in easing the IPv6 network renumbering process.

2. Prior Art and Available tools for Network Renumbering (Southampton)

In this section we review prior work undertaken in the IPv4 renumbering space, and IPv6 specific considerations and tools to support IPv6 renumbering. This represents the set of 'thinking' and tools brought to this activity in the project.

Prior work in the IPv4 space has been relatively limited due to the relatively easily obtainable Provider Independent (PI) address space for IPv4. Although there is currently a proposal in front of ARIN for limited PI space for IPv6 (on the basis the recipient could qualify for an ASN by the Regional Internet Registry's rules), there is no indication at present that PI address space will become generally available for IPv6 enterprise sites.

As a result, there are also few tools in the arsenal of a network administrator who needs to renumber a network. The process has to be carried out almost entirely manually with little scope for automation. While IPv6 offers some potential advantages in this space (described later in this section), these are as yet largely unproven in testbeds, yet alone production deployments. Some tools that may have been available (in particular A6 DNS records) have been deprecated by the IETF in the interests of simplifying IPv6 deployment). It is unlikely that there will ever be a magic 'Renumber Now' button for IPv6 networks, in this document we attempt to take steps towards easing the process, by identifying the scenarios, issues and tools that can and could be used, and running experiments in required underlying protocol implementations (e.g. RFC3484 address selection).

Tools to test the renumbering process themselvcs are also limited. The TAHI conformance suite does not appear to have specific tests for renumbering, rather it allows specific IPv6 protocol operations to be verified (i.e. more 'individual' actions, not a sequence (as required by [BAKER]).

2.1. The IETF PIER WG

A number of years ago (1996-1997), the Procedures for Internet/Enterprise Renumbering (PIER) WG spent time considering the issues for IPv4 renumbering. The WG produced three RFC documents. In RFC1916 [RFC1916], a "call to arms" for input on renumbering techniques was made. RFC2071 [RFC2071] documents why IPv4 renumbering is required. Interestingly, many, but not all, of the drivers have changed with respect to IPv6. In RFC2072 [RFC2072], a Router Renumbering Guide, some operational procedures are given, much as they are in Baker [BAKER] for IPv6.

Reflection on RFC2071 is interesting, witness the quote: "It is also envisioned that Network Address Translation (NAT) devices will be developed to assist in the IPv4 to IPv6 transition, or perhaps supplant the need to renumber the majority of interior networks altogether, but that is beyond the scope of this document." That need however is still very strong, particularly given the lack of Provider Independent (PI) address space in IPv6 (in IPv4, PI address space exists mainly for historical, pre-CIDR reasons).

RFC2072 is more interesting in the context of this document. Some is certainly relevant, though much is not, due to the inherent changes in IPv6. For example, there is no CIDR and address aggregation is given as mandate. Also, IPv6 subnets are in effect fixed length (/64), so local administrators do not need to resize subnets to maximise efficient use of address space as they do in IPv4.

One core message from RFC2072 that holds true today is the observation is made that renumbering networks whilst remaining the same hierarchy of subnets (i.e. the cardinality of the set of prefixes to renumber remains constant) is the 'easiest' scenario to renumber; when each "old" prefix can be mapped to a single "new" prefix.

A distinction of this work is that, where the PIER working group consider the transition from IPv4 to IPv6 addressing as a renumbering scenario, we strictly consider only the renumbering from IPv6

prefixes to other IPv6 prefixes and leave transition to well documented techniques such as those from the PIER working group.

2.2. IETF Procedures for Network Renumbering without a Flag Day

The IETF v6ops WG has defined a process for IPv6 network renumbering without a flag day [BAKER]. The method describes a phased approach to the transition from the old to the new prefix, which utilises multi-addressing during the transition such that nodes are temporarily connected using both prefixes.

The procedure loosely follows the following methodology:

- Stable use of the existing prefix
- Preparations for introducing the new prefix, inlcuding DNS
- Configuring network elements for the new prefix
- Adding new host addresses
- Stable use of both network prefixes
- Transition from the old to the new prefix, including DNS
- Removing the old prefix
- Stable use of (only) the new prefix

The experiments described later in this text apply this procedures in different contexts (SOHO and backbone networks). At present, this text has to be considered best practice in IPv6 network renumbering, until operational experience validates this is shows otherwise.

2.3. IPv6 Protocol Features and their Effects on Renumbering

IPv6 includes a number of notable features that can help or hinder - and sometimes both - renumbering episodes. This section discusses these features and their associated effects for consideration when undertaking network renumbering, both in terms of how they can be used to ease the process, as well as potential pitfalls that should be considered.

2.3.1. Multi-addressing

As per RFC3513 [RFC3513], IPv6 hosts may be multi-addressed. This means that multiple unicast addresses can be assigned and active on the same interface. These addresses can have different reachabilities ('scopes' such as link-local or global), different statuses including 'preferred' and 'deprecated', and may be ephemeral in nature (such as care-of addresses when attached to a foreign network [RFC3775] or IPv6 Privacy addresses [RFC3041]). RFC3484 address selection semantics [RFC3484] determine which of the "MxN" address pairs to use for communication in the general case.

During a renumbering episode, the addition of an extra address for an endpoint increases the number of possible source-destination address pairs for communications between nodes to use. The address selection mechanisms specified by RFC3484 are currently at varying stages of implementation in operating systems.

RFC3484 also specifies policy hooks to allow administrative override of the default address selection behaviour, for example to specifically lock-down a source prefix to select for a set of particular destinations. Where this policy-based address selection was designed for transition from IPv4 and early IPv6 multi-homing considerations, it is perceived likely to be of benefit to bespoke renumbering tool development.

2.3.2. Multi-homing techniques

A multi-homed site is a site which has multiple upstream providers. A site may be multi-homed for various reasons, however the most common are to provide redundancy in case of failure, to increase bandwidth, and to provide more varied, optimal routes for certain destinations.

In renumbering, multi-homing will either be a temporary state, during the transition, or be a permanent feature of the network configuration, which may be being altered during the renumbering.

2.3.2.1. Relevance of multi-homing to renumbering

As discussed in [BAKER], during the 'without a flag day' renumbering procedure there will be a period where both the old and the new prefixes are stable and valid for the network. During such a period, the network will therefore be multi-homed, and as such many of the issues relating to multi-homing in IPv6 are also relevant, albeit in a small capacity, to the renumbering procedure. A stable multi-homed situation must therefore be a requirement for renumbering without a 'flag day'.

In such a situation, however, the multi-homed state will not be permanent, and will only exist for the duration for which it is required, i.e. for the period during the renumbering procedure when both prefixes should be valid.

Renumbering can also occur, however, in a network that is already multi-homed, for example with redundant links to multiple providers. Such a site may wish to renumber for any of the situations given in the earlier section, as well as renumbering because of changes in the number of upstream providers. If at least one of the upstream links remains unchanged during the renumbering, however, then these links could be used exclusively for that period, alleviating some of the issues with prefix changes. Until the best practice for the multi-homing situation is defined, however, its effect on renumbering is not a focus of this document.

2.3.2.2. Current situation with IPv6 multi-homing

Unlike IPv4 multi-homing, where PI address space is relatively easy to obtain and thus a site can broadcast its own routing information, most IPv6 addresses will be PA addresses and thus the site will have no control over routing information. Multi-homing in IPv6 therefore does not necessarily exist in the same way as in IPv4 and the multi6 [MULTI6] working group has completed its charter in a first step to try to find a solution, to be carried on by the shim6 IETF WG. Current solutions [HUSTON] examine the potential for using identifiers within IP to identify a host, as opposed to an IP address, so that connections can continue unhindered across renumbering events. Such solutions are however very much in their infancy and as yet do not provide a stable solution to this problem.

Support for the level of multi-homing required during a renumbering exercise is however mostly provided by multi-addressing, since all that is primarily required is stable use of either prefix for a given session.

2.3.3. Mobile IPv6

Mobile IPv6 (MIPv6) [RFC3775] specifies routing support to permit an IPv6 host to continue using its "permanent" home address as it moves around the Internet. Mobile IPv6 supports transparency

above the IP layer, including maintenance of active TCP connections and UDP port bindings. There are a number of issues to take into account when renumbering episodes occur where Mobile IPv6 is deployed.

Renumbering a network which has mobile IPv6 active is a potentially complex issue to think about. In particular, can changed router advertisements correctly reach the mobile nodes, and can they be correctly renumbered, like a node on the local network? In addition, an even more complex issue is what happens when the home agent renumbers? Is it possible for the mobile nodes to be informed and correctly renumber and continue, or will the link be irretrievably broken?

2.3.3.1. Visited site renumbers when mobile

When a node is mobile and attached to a foreign network it, like any other node on the link, is subject to prefix renumbering at that site. Detecting a new prefix through the receipt of router advertisements, the mobile node can then re-bind with its home agent informing it of its care-of address - just as if it had detached from the foreign network and migrated elsewhere. Where the node receives forewarning of the renumbering episode, the Mobility specification suggests that the node explicitly solicits an update of the prefix information on its home network

2.3.3.2. Home site renumbers when mobile

When mobile, a host can still be contacted at its original (home) address. Should the home network renumber whilst the node is away but active (i.e. having bound to the home agent and registered a live care-of address), then it can be informed of the new global routing prefix used at the home site through the Mobile Prefix Solicitation and Mobile Prefix Advertisement ICMPv6 messages (sections 6.7 and 6.8 of RFC3775 respectively).

2.3.3.3. Home site renumbers when disconnected

Finally, if a mobile node is detached (i.e. no binding with the home agent exists with the node present on a foreign network) and the home network renumbers, the recommended procedure - documented as an appendix to the mobility specification and therefore not necessarily proven - is to fall back to alternative methods of 'rediscovering' its home network, using the DNS to find the new global routing prefix for the home network and therefore the Home Agent's subnet anycast address, 'guessing' at what the node's new home address would be on the basis of a 64 bit prefix and 64 bit interface identifier, and then attempting to perform registration to bind its new location.

2.3.4. Multicast

IPv6 supports an enriched model of multicast compared to IPv4 in that there are well-defined scopes for multicast communication that are readily expressed in the protocol's addressing architecture. Multicast features much more prominently in the core specification, for example it is the enabling technology for the Neighbour Discovery protocol (a much more efficient approach to layer 2 address discovery than compared to ARP with IPv4).

Where multicast is used to discover the availability of core services (e.g. all DHCPv6 servers in a site will join FF05::1:3), the effect of renumbering the unicast address of those services will mean that the services are still readily discoverable without resorting to a (bespoke or otherwise) service location protocol to continue to function - particularly if (unicast) ULAs are not deployed locally.

One issue related to IPv6 multicast and renumbering is the embedding of unicast addresses into multicast addresses specified in RFC3306 [20] and the embedded-RP (Rendezvous Point) in RFC3956 [21]. The former is purely a way of assigning addresses that helps with multicast

address assignment, avoiding different sites from using the same multicast addresses. If a site's unicast prefix changes, then one will also need to change the multicast addresses. By way of example, a site renumbering away from prefix 2001:DB8:BEEF::/48" might have globally-scoped multicast addresses in use under the prefix "FF3E:30:2001:DB8:BEEF::/96". One may continue using the old addresses for a while, but this should be avoided since another site may inherit the prefix and they may end up using the same multicast addresses.

The issue with embedded-RP is that, by definition, the RP address is embedded. So if the RP address changes, then the group addresses must also be changed. This may happen not only when a site is renumbered, but also if a site is restructured or the RP is moved within the site. The embedded address is used by routers to determine the RP address. Applications must use new group addresses once the RP is not available on the old address.

Another interesting topic is multicast source renumbering. With traditional multicast a source should be able to start streaming from a new address, and nodes belonging to the multicast group will immediately start receiving. There might be some application issues though. If sources are identified by the source address only, then this might appear as a new source to the receivers (as they would where IPv6 Privacy addresses are used). Using RTP a receiver may determine it's the same source.

With Source Specific Multicast (SSM), source renumbering is more complicated since receivers must specify exactly which sources they want to receive from. This means that receivers must somehow be told to join the new source addresses, and must be able to discover those addresses.

2.3.5. Unique Local Addressing

[ULA] suggests that the use of Local IPv6 addresses in a site results in making communication using these addresses independent of renumbering a site's provider based global addresses. It also points out that a renumbering episode is not triggered when merging multiple sites that have deployed centrally assigned unique local addresses [ULA-C] because the FC00::/8 ULA prefix assures global uniqueness.

When merging two sites that have both deployed FD00::/8 locally assigned ULA prefixes, the chance of collision is inherently small given the pseudo-random global-ID determination algorithm of [ULA]. Consideration of possible collisions may be prudent however unlikely the occurrence may be.

With reference to [BAKER], the adoption of ULA to assist in network renumbering can be considered a 'seasoning' of Baker's renumbering procedure: where interaction between local nodes and their services cannot suffer the inherent issues observed when migrating to a new aggregatable global unicast prefix, the use of FC00::/7 unique local addresses may offer an appropriately stable and reliable solution. The use of ULAs in single-site networks (e.g. SOHO) appears straightforward and of immediate benefit with regards renumbering episodes triggered by uplink connectivity changes. How their use scales to multi-site (e.g. Enterprise or use in ISP or transit networks) is not so evident.

If addresses under a global routing prefix are also deployed, then nodes will need to cater for being multi-addressed, e.g. follow the principles laid out in RFC3484. The administrator should ideally be able to set local policy such that nodes use ULAs for intranet communications and global addresses for global Internet communications. The use of ULAs internally would in principle mitigate against global address renumbering of nodes. One cost of such dual-scope deployment is the implied requirement to run a 'two-faced' DNS, which can add an extra form of complexity.

The use of ULAs may not necessarily be accompanied by PA addresses. If addresses under a PA global routing prefix are not used, some form of IPv6 NAT or application layer gateway deployment will be required for ULA-only nodes internal to the network to communicate with external nodes that are not part of the same ULA topology, that is destination nodes that are not part of the same administrative domain from which the ULA allocation of the local node is made, nor part of a predetermined routing agreement between two organisations utilising different ULAs for nodes within their own sites. ULAs are not intended to be routed globally. However, use of IPv6 with NAT destroys the very advantage of IPv6 deployment, thus it is not recommended.

ULAs appear to lend themselves particularly well for long-lived sessions (from the categorisation Section 4.2.3) whose nature is intra-site, for example local filestore mounts over TCP-mounted NFS: With clients using ULA source addresses to mount filestore using the ULA of an NFS server, both client and server can have their global routing prefix renumbered without consequence to ongoing local connections.

2.3.6. Anycast addressing

Syntactically indistinguishable from unicast addresses, 'anycast' offers nodes a mean to route traffic toward the topologically nearest instance of a service (as represented by an IP address), relying on the routing infrastructure to deliver appropriately. RFC2526 [RFC2526] defines a set of reserved subnet anycast addresses within the highest 128 values of the 64 bit IID space. Of that space, currently only three are used, of which one is actively used and is for discovery of Mobile IPv6 Home-Agents. At the current time there are no 'global' well-known anycast addresses assigned by IANA.

In order to participate using anycast, nodes need to be configured as routers (to comply with RFC3513) and exchange routing information about the reachability of the specific anycast address. This extra level of administration requirement is negligible in the context of services as the services themselves would need configuration anyway.

There have been proposals to define globally well-known anycast addresses for core services, such as the DNS [DNSRES]. Anycast scales with regard subnet-anycast in the sense that the global routing prefix used to direct packets to an anycast node within a site is no different from any other host, and therefore nothing 'special' in the global routing architecture is required - only locally within the site does the multi-node nature of anycast need to be considered.

However, for global well-known anycast addresses to be defined, host-specific routes will need to be advertised and distributed throughout the entire Internet. As acknowledged in [RFC3513], this presents a severe scaling limit and it is expected that support for global anycast sets may be unavailable or very restricted. A good discussion of best current practice for service provision using anycast addressing can be found in [ANYOP].

The use of well-known anycast addresses would assist the renumbering exercise by removing the requirement to change the addresses in the configuration of such services. The use of anycast DNS would alleviate concerns with ensuring node reconfiguration, for example when using Stateless DHCPv6. While anycasting datagram-based services such as DNS pose little problems, anycast does not maintain state, and so it would not be guaranteed that sequential TCP packets were to go to the same host. As discussed in [RFC1546], responses from TCP sessions begun to an anycast address should be sent from the unicast address, and future communication should continue with this address. While this means that communication will continue with the same unicast address, that address is subject to the standard address deprecation and validity. Note that anycasting of this form can be an alternative to site or organisational scope multicast service discovery.

2.4. Node Configuration Issues

This section discusses how IPv6 node configuration protocols (both stateless and stateful, including DHCPv6, as well as ICMP router renumbering messages) can be used to facilitate a renumbering event, plus any complications caused by these processes, to which consideration should be given.

2.4.1. Stateless Address Autoconfiguration

Many IPv6 networks are likely to be configured using Stateless Address AutoConfiguration [RFC2462] (SLAAC), and in order to work through the multi-staged process as documented by Baker [1], the new prefix is introduced via router advertisements, and then the old prefix is deprecated, and finally removed.

Initially the router advertisements will contain only the prefix of the old network, then for a time they will contain both the old and the new, but with a shorter (zero) lifetime on the old prefix to indicate that it is deprecated. Finally the router advertisements will contain only the new prefix.

2.4.1.1. Router Advertisement Lifetimes

RFC2462 (IPv6 Stateless Autoconfiguration) specifies the technique for expiring assigned prefixes and then invalidating them, such that a network has opportunity to gracefully withdraw a prefix from service whilst not terminally disrupting on-going applications that use addresses under it. Section 5.5.4 of RFC2462 in particular details the procedure for deprecation and subsequent invalidation.

By mandating as a node requirement the ability to phase out addresses assigned to an interface, network renumbering is readily facilitated: subnet routers update the pre-existing prefix and mark them as 'deprecated' with a scheduled time for expiration and then advertise (when appropriate) the new prefix that should be chosen for all outgoing communications.

2.4.1.2. Stateless Configuration with DHCPv6

Sometimes, DHCPv6 will be used alongside SLAAC. SLAAC will provide the address assignment, and DHCPv6 will provide additional host configuration options, such as DNS servers. If any of the DHCPv6 options are directly related to the IPv6 addresses being renumbered, then the configuration must be changed at the appropriate time during the renumbering event, even though it itself does not handle the address assignments.

Since the configuration is stateless, the DHCPv6 server will not know which clients to contact to inform them to refresh. Clients of the configuration protocol should poll the service to obtain potentially updated ancillary data, such as suggested by [DHCLIFE]. It is proposed that a new DHCPv6 service option is added to inform clients of an upper bound for how long they should wait before re-requesting service information.

2.4.2. Stateful Configuration with DHCPv6

As opposed to stateless autoconfiguration, IPv6 stateful or managed configuration can be achieved through the deployment of DHCPv6. Section 18.1.8 of [RFC3315] details how a node should respond to the receipt of stateful configuration data from a DHCPv6 server where the lifetime indicated has expired (is zero). Section 19.4.1 details how clients should respond to being instructed by DHCPv6 servers to reconfigure (potentially forceful renumbering). Section 22.6 details how prefix validity time is conveyed (c.f. the equivalent data in SLAAC's Router Advertisement).

In order to renumber such a network, the DHCPv6 server should send reconfigure messages to inform the clients that the configuration has changed, and the clients should re-request configuration details from the DHCPv6 server. This, of course, relies on the clients correctly responding to such messages.

Where DHCPv6 has been employed, careful consideration about the configuration of the service is required such that administrators can be confident that clients will re-contact the service to refresh their configuration data. As alluded to in sections 22.4 and 22.5 of [RFC3315], the configurable timers that offer servers the ability to control when clients re-contacts the server about its configuration can be set such that clients rarely (if ever) connect to validate their configuration set.

The approach described in [DHCLIFE] allows the lifetime of other configuration information supplied by DHCPv6 to be ramped down in preparation for a planned renumbering event.

2.4.2.1. Prefix Delegation

Where stateless autoconfiguration enables hosts to request prefixes from link-attached routers, prefix delegation enables routers to request a prefix for advertising from superior routers, i.e. routers closer to the top of the prefix hierarchy - typically topologically closer, therefore, to the provider. Once the router has been delegated prefix(es), it can begin advertising it to the connected subnet (perhaps even multi-link) with indicators for hosts to use stateful (DHCPv6) or stateless address autoconfiguration as per RFC2461.

There have been two principal approaches to prefix delegation proposed: HPD (Hierarchical Prefix Delegation for IPv6), which proposed the use of bespoke ICMPv6 messages for prefix delegation, and IPv6 Prefix Options for Dynamic Host Configuration Protocol [RFC3633], which defines a DHCPv6 option type. Of the two approaches, the DHCPv6-based approach has received wide support and is on the standards track.

2.4.2.2. Source Address Selection Policy distribution

It has been proposed that DHCPv6 could also be used to distribute source address selection policy to nodes [SRCPOL]. The model proposes that consumer edge router receives policies (e.g. from multiple ISPs in the case of multi-homed networks) and re-distributes them to end nodes. The end nodes then put them into their local policy table, which leads to appropriate source address selection. Where the design goal was a distribution mechanism in light of multi-homed networks, the adoption of the technique for the multi-prefix states of [BAKER] during renumbering appears appropriate.

2.4.3. Router Renumbering

RFC2894 [RFC2894] defines a mechanism for renumbering IPv6 routers throughout a network using a bespoke ICMP message type for manipulating the set of prefixes deployed throughout subnets. Through the use of prefix matching and a rudimentary algebra for bit-wise manipulation of prefix data bound to router interfaces, the mechanism enables administrators to affect every router within a scope from a single administration workstation. One drawback of RFC2894 is that it requires an enterprise-wide IPsec infrastructure to be deployed to secure the ICMP messages in order to be compliant.

The approach utilises multicast communication to the all-routers address, FF05::2, scoped to the entire 'site' as determined by router filter policy to distribute configuration updates to all (compliant) routers. The mechanism also works with more specific addressing modalities, such as link-local multicast (FF02::2) to reach all routers on a specific link, or directed unicast to affect a specific router instance. When surveying current implementations very few IPv6 implementations bound

their interfaces to the Site-wide All-Routers multicast address (FF05::2), and fewer still have implementations of RFC2894.

Example use cases cited in RFC2894 are for deploying global routing prefixes across a hierarchical network where site-locals already exist (presumably updated now to Unique Local Addresses), and for renumbering from an existing prefix to another in a similar manner to that proposed by Baker (i.e. the deployment of a new prefix alongside the existing one, which is deprecated and subsequently expired and removed - using the same mechanism described).

The specification was developed before the shift in recommendation away from the Top-, Next- at Site-Level Aggregation Identifier address allocation hierarchy of RFC3513, although the techniques documented for renumbering the global routing prefix and subnet ID components in the updated address allocation recommendations are not affected by the architectural change.

As with other prefix assignment techniques, it is the responsibility of the node to correctly deprecate and then expire the use of a previously assigned prefix as defined by the IPv6 Neighbour Discovery protocol, RFC2461 [RFC2461], section 4.6.2 describing the Prefix Information option in particular.

3. Scenarios and Considerations for IPv6 Renumbering (Southampton)

In this section we describe scenarios that may trigger a renumbering event, and considerations that may affect that renumbering event.

3.1. Renumbering Event Triggers

This section details typical actions that result in the need for a renumbering event, and thus define the scenarios for renumbering. In many instances, in particular those where no "flag day" is involved, the process of renumbering will inevitably lead to a scenario where hosts are multi-addressed or multi-homed as one phase of the renumbering procedure. The relationship between renumbering and multi-homing is discussed in this document.

In other instances, e.g. a change in the IPv4 address offered by a provider to a site using 6to4 [RFC3056], the change offers no overlap in external connectivity or addressing, and thus there is no multi-homing overlap.

Triggers may be provider-initiated or customer-initiated.

Triggers and scenarios for IPv4 renumbering are discussed in RFC2071, but many of these are no longer relevant, and in IPv6 some new triggers exist, e.g. those related to network mobility or IPv6 transition tools.

3.1.1. Change of uplink prefix

One of the most common causes for renumbering will be a change in the site's upstream provider. As per RFC3177 [RFC3177], the typical allocation for an IPv6 site is a /48 size prefix taken from the globally aggregated address space of the site's provider. With IPv6, sites are highly unlikely to be able to obtain provider independent (PI) address space, as have in some cases been obtained in the past with IPv4. Rather, sites use provider assigned (PA) addressing. As a result, if a site changes provider, it must also change its IPv6 PA prefix.

3.1.1.1. Migration to new provider

In the simplest case, the customer is triggering the renumbering by choosing to change the site's upstream provider to a new ISP and thus a new PA IPv6 prefix range. This may simply be in the form of selecting a new commercial provider, although there are several other possible scenarios, such as changing from a dial-up to a broadband connection, or moving from a community wireless connection to a fixed broadband connection.

3.1.1.2. Dial on Demand

A site may connect intermittently to its upstream provider. In such cases the prefix allocated by the provider may change with each connection, as it often does in the case of single IPv4 address allocations to SOHO customers today. Thus the site may receive a prefix still in its provider PA range, but the prefix may vary with each connection, causing a renumbering event.

Dynamically assigned IP addresses are common today with dial-up and ISDN Internet connections, and to a lesser extent some broadband products, particularly cable modems. Usually with dynamically assigned IP addresses on broadband products, the address is only likely to change when the customer reconnects, which could be very infrequently.

This case can be mitigated by encouraging ISPs to offer static IPv6 prefixes to customers. Where /48 prefixes are provided, a large ISP may be forced to require significantly more than the "default" /32 allocation from an RIR to an ISP to be able to service its present and future customer base. With always-on more common in new deployments, provider re-allocation should be less common; however the practice of reallocating IPv4 addresses in SOHO broadband networks is not uncommon in current broadband ISPs.

3.1.1.3. Provider migration and upstream renumbering

A site's upstream provider may need to renumber, due for example to a change in its network topology or the need to migrate to a different or additional prefix from its Regional Internet Registry (RIR). This will in turn trigger the renumbering of the end site.

Such renumbering events would be expected to be rare, but it should be noted that RIR-assigned IPv6 address space is not owned by an ISP.

3.1.1.4. IPv6 transition

During transition to IPv6, there are several scenarios where a site may have to renumber. For example, if the site uses 6to4 for access and its IPv4 address is dynamically assigned, an IPv6 renumbering event will be triggered when the site's IPv4 address changes.

Another likely renumbering event would be the change of transition mechanism, such as from 6to4 to a static IPv6-in-IPv4 tunnel, or from any one of those mechanisms to a native IPv6 link. When changing from 6to4 (2002::/16) addresses to native global aggregatable unicast addresses, renumbering would be unavoidable. When migrating from a tunnelled to a native connection, renumbering may not be necessary if the same prefix can be routed natively, however this would be provider-dependent.

In addition, there are likely to be many cases of network renumbering occurring when the old 6bone prefix (3FFE::/16) is phased out as per RFC3701 [RFC3701], and networks still using it will have to renumber.

Finally, there is at least one transition mechanism, ISATAP [ISATAP], that uses specially crafted host EUI-64 format addresses. Should a site migrate from ISATAP to use either conventional EUI-

64 addressing (via stateless address autoconfiguration or perhaps DHCPv6), then renumbering would be required at least in the host part of addresses.

It is also worth noting that nodes that use IPv6 Privacy Extensions [RFC3041] will in effect renumber the host part of their address on a frequent basis, in the case of one popular implementation on a daily basis if the node remains on-link on the same network.

3.1.2. Change of internal topology

A site may need to renumber all or part of its internal network due to a change of topology, such as creating more or less specific subnets, or acquiring a larger IPv6 address allocation. Motivations for splitting a link into separate subnets may to meet security demands on a particular link (policy for link-based access control rules), or for link load management by shuffling popular services to more appropriate locations in the local topology. Link-merging may be due to department restructuring within the hosting organisation, for example.

3.1.3. Acquisition or merger

Two networks may need to merge to one due to the acquisition or merger of two organisations or companies. Such a reorganisation may require one or more parts of the new network to renumber to the primary PA IPv6 prefix.

3.1.4. Network growth

A site that is allocated a /48 prefix may grow to a size where it needs to use a larger prefix for internal networking. Sites in the early stages of IPv6 deployment may only request a /48, even if they are likely to outgrow such a prefix in time. In such a case site-wide renumbering may be required to utilise the new prefix if organisational restructuring also happens due to the growth.

3.1.5. Network mobility

This covers various cases of network mobility, where a static or nomadic network may obtain different uplink connectivity over time, and thus be assigned different IPv6 PA prefixes as the topology changes. One example is the "traditional" NEMO network [NEMO], another may be a community wireless network where different sets of nodes gain uplink connectivity - typically to the same provider - at different times.

3.2. Renumbering Requirements

In this section we enumerate potential specific goals or requirements for sites or users undergoing an IPv6 renumbering event.

3.2.1. Minimal disruption

The renumbering event should cause minimal disruption to the routine operation of the network being renumbered, and the users of that network.

Disruption is a difficult term to quantify in a generic way, but it can be expressed by factors such as:

- Application sessions being terminated
- Security controls (e.g. ACLs) blocking access to legitimate resources
- Unreachability of nodes or networks

- Name resolution, directory and configuration services providing invalid (out-of-date) address data
- Limitation of network management visibility

These disruptive elements will be covered in situ as we discuss protocol features and other renumbering considerations later in this memo.

3.2.2. Session survivability

The concept of session survivability is catered for by [BAKER] in that new sessions adopt either old or new prefix based on the state of the renumbering process, as discussed in Section 5.1. However, other approaches to renumbering networks may be appropriate in certain deployments, such as where "flag days" are unavoidable, such as where two live prefixes are being "swapped". In these cases, further consideration for existing sessions (their longevity, frequency, independence across interactions, etc.) is required.

Some protocols are specifically geared to aid session survivability, e.g. the Stream Control Transmission Protocol (SCTP) [RFC2960], and may prove valuable in mission-critical renumbering scenarios, in particular the extension that enables the dynamic addition and removal of IP addresses from an SCTP endpoint association [SCTPDAR].

Sessions may be administratively maintained, such as NFS mounts for user filestore, or they may be user-driven, e.g. long-running ssh sessions.

In general, it is important to consider how TCP and the applications above it handle the connection failures that may result from a change in address.

There are different classes of session duration, as described in the following sections.

3.2.2.1. Short-term session survivability

A typical short-term session would involve a request-response protocol, such as HTTP, where a new network connection is initiated per transaction, or at worst for a small transaction set. In such cases the migration to a new network prefix is transparent: the client can use the new prefix in new transactions without consequence. Some applications, however, may be skewed by such a shift in connection source for the same entity 'user', for example applications that use recent connection history as a cue to identity (e.g. POP-before-SMTP[1] as used by many dial-on-demand ISP customers), or for applications that care about connection statistics (the same user web-browsing "session" may be split into two where a renumbering event occurs in-between client transactions).

3.2.2.2. Medium-term session survivability

A medium-term session is typified by an application or service that may persist for perhaps a period of a few minutes up to a period of a day or so. This might involve a TCP-based application that is left running during a working day, such as an interactive shell (SSH) or a large file download.

3.2.2.3. Long-term session survivability

Long term sessions may typically run for several days, if not weeks or months. These might typically include TCP-based NFS mounts, or long-running TCP applications. Sessions in this context may also include those applications that, once started, do not re-resolve names and so repeatedly open new connections or send new datagrams to the same (as bound at time of initialisation) address throughout their execution lifetime. Even if at API-level applications do

[1] http://popbsmtp.sourceforge.net/

attempt to re-resolve the symbol to which they desire to connect, the behaviour of the resolvers is unclear as to whether mappings are refreshed from the naming service, and as such even if the renumbering site does update its DNS (or NIS, LDAP database etc.), the local result may indeed be cached without any indication passed back up to the application as to how 'old' said binding information is.

3.2.2.4. "Sessions" in non-session based transports

UDP transport protocols, such as UDP-based NFS mounts, maintain the status of a 'session' by keeping state at one or both ends of the communication, but without a persistent open socket connection at the network layer. If, due to node renumbering, one endpoint changes address then that state becomes invalid and the 'session' interrupted.

IP addresses are also seen carried in higher-layer protocols, e.g. application sessions, such as with FTP. Any application that makes use of layer-3 address data as a unique end-point identifying token may be disrupted by the address of the node changing to which that token relates. This may not be an issue in cases where the token is treated as abstract (i.e. literally just a token), however where locator semantics are inferred, subsequent attempts to 'resolve' the token to an address endpoint for communication, for example, will fail.

3.3. Administrative Considerations for Renumbering

This section is concerned with factors that affect the renumbering procedure, from a network administration viewpoint. In particular, this section discusses areas that a network administrator should consider before undertaking a renumbering event, to ensure that it proceeds smoothly. This includes considerations of event frequency, scalability, and those relating to delays in information propagation.

3.3.1. Router Advertisement Lifetimes

IPv6 Stateless Autoconfiguration allows the expiration of assigned prefixes. This process permits existing sessions to continue while preferring a new prefix. It should be noted, however, that there are some limitations in the specification that have an impact in renumbering. In particular, it is not possible to reduce a prefix's lifetime to below two hours if it has previously been available at a longer validity. This therefore emphasises the need to plan renumbering events in advance if at all possible, to reduce the lifetime as required, within these limitations.

3.3.2. Border filtering

Multi-addressing allows multiple globally reachable addresses to be assigned to node interfaces, but one administrative caveat that arises is that of site ingress filtering: not only is it the norm for sites to filter at their border router traffic that is not destined to local subnets, but it is also increasingly common for sites to filter on egress to prevent administratively local addresses (such as the, now deprecated, site-local prefix) 'leaking' traffic or for mis-configured hosts (e.g. visitors with manually configured stacks without Mobile IPv6) from sourcing traffic that cannot be routed back (cases of which may include deliberate IP spoofing or DDoS attempts).

3.3.3. Frequency of renumbering episodes

The many different renumbering scenarios can have vastly different frequencies of renumbering events. In the case of a provider offering only dynamically assigned IP addresses, it could be very frequent, for example as frequent as 'per-connection' for dial-on-demand services, or weekly for

some broadband services. Such renumbering events usually only occur when a customer reconnects to such services or are explicitly cited in a subscription agreement and as such are often pre-determined.

The renumbering of a site due to upstream renumbering is relevant to all connections from a small dial-up link to a large enterprise. It is of particular interest since the end user has no control over the timing or frequency of the renumbering events. It is expected, however, that such events are likely to be very infrequent.

The other irregular renumbering events are those that occur due to end user migrating, either to a new provider, or to a new address allocation of their choosing. The timing of such an event is therefore often within the control of the end user (within reason), and are also likely to be one-off events, or at the very least, highly infrequent.

3.3.4. Delay-related Considerations

When considering a renumbering event, both the planning of, and responses to the event are affected by temporal factors. The amount of time available in which to undertake the operation can change the administrative actions required, and this section aims to discuss some of these issues.

3.3.4.1. With or without a flag day

A network may be renumbered with or without a flag day. In the context of this document we are focusing on without a flag day, although many of the issues will still apply when renumbering is effected with a flag day.

Despite the similarities, because there is an outage of services when renumbering with a flag day, it is not necessary to ensure continuity of network connections, and almost all reconfiguration can be done during the outage, thus greatly simplifying the task of renumbering.

3.3.4.2. Freshness of service data

One of the largest issues when renumbering a network will be the effect on applications that are already running. In particular, applications that periodically contact a particular host may do an initial hostname lookup, and cache the result for use throughout the lifetime of the program. In such a situation, there is no way for the application to find out that the host in question has been renumbered, and it should stop using its already cached address. It is therefore recommended that applications should regularly request hostname lookups for the desired hosts, leaving the caching to the resolver. It is then up to the resolver to ensure that resource record TTLs are observed, and its cached response is updated as necessary.

Despite this, there is still a serious issue in that there is no method of caching resolvers knowing when a renumbering event is going to take place. If a typical RR's TTL is one day, then that should be reduced not less than a day before the renumbering event, so that resolvers will more frequently check for changed records. This will work successfully for a pre-planned renumbering event, but problems of stale, cached records will exist if the renumbering event is unplanned (e.g. by receiving a new router advertisement from upstream).

There are also cases where the use of a resolver is not practical, such as with packet filter rules. If a packet filter has been configured with explicit hostnames, these are translated to IP addresses for fast packet matching. The per-packet resolver function is highly undesirable from a pure performance perspective. Such a packet filter is likely to need to be reloaded for the DNS changes to be recognised.

A similar problem exists when a nameserver is renumbered. If the operating system's resolver has cached the nameserver address, it will at some point find it unavailable. To mitigate this problem, it is suggested that at least one off-site nameserver is included in the configuration. In addition, well-known anycast addresses could be used, so that the client's DNS configuration does not need to be changed at all during the renumbering event.

The basic process of renumbering, involving the introduction of a new prefix and the deprecation and eventual removal of the old prefix, could be hypothetically handled by a special tool, with no manual intervention. Such a tool would have to become significantly more complex in order to handle all the cases where IP addresses are explicitly specified.

Other particularly notable cases that could be changed with a tool, were it to be developed, include DNS zone files and DHCPv6 configuration. Deployment of such a tool, even if possible, would be made complex through the requirement to authenticate the updates to each instance of the deployed literals.

3.3.4.3. Availability of old prefix

The duration of the period where the old prefix remains available affects the length of time that can be allowed for the renumbering procedure, and the maximum time for which existing sessions could continue. If end users have control over the renumbering procedure (such as when changing provider), then they can continue providing the old prefix for as long as required, within reason (such as cost aspects). This heavily mitigates the issues of session survivability, and relaxes the speed at which hosts must be reconfigured.

If the end users do not have such control, such as when the upstream provider forces the renumbering, the availability of the old prefix is determined entirely by the upstream provider's willingness to continue providing it, which is likely to be based on the technicalities of their own renumbering situation. The end user should therefore not rely on retaining the old prefix for a relatively long period of time. In addition, many situations, such as dial-on-demand with dynamic IP addresses, and nomadic networks, will lose their old prefix quickly, if not almost instantaneously.

It would be possible to continue using the old prefix internally, even when the external connectivity for that prefix is no longer active, for example to keep access to core services such as DNS servers while the transition is taking place. This should, however, be considered bad practice in case of route leaking and application confusion, as well as preventing access to the addresses if they have been reassigned, and as such this should only be used as a last resort to ensure internal continuity of service, if the availability of the old prefix is too short to allow a full transition to take place.

3.3.4.4. Duration of overlap

A key operational decision when renumbering is enforced due to a change in connectivity provider is how long to sustain the overlap of two live prefixes. The trade-off to be made is the cost of maintaining two contracts with separate providers against the 'smoothness' of the transition to the new prefix as regards local administration overheads, service migration, etc. Where larger corporations can likely suffer the increased financial costs, SMEs and SOHOs might consider as little as one month's overlap too expensive, and so Baker's State 5 (Stable use of either prefix) [BAKER] is unattainable in such scenarios.

In some cases, there may be technical reasons for the overlap to not be feasible, such as with xDSL provision where the new service is a drop-in replacement for the old and the two cannot co-exist (for example, because the provision of the service requires the whole circuit resource from exchange to customer).

3.3.5. Scalability issues

During the renumbering transition, there will be a time when two prefixes are valid for use. At this point, there will be a considerable amount of configuration that will have to be (temporarily) duplicated. In particular, routing entries on the hosts will be doubled, and there will, for a short period, be two forward DNS records for every hostname. Security is another key scalability issue. All access control lists, packet filters, etc, will need to be updated to cope with the multiple addresses that each host will have. This could have a noticeable impact on packet filter performance, especially if it leads to, for example, the doubling of several hundred firewall rules.

The scalability issues created by the increase in configuration to cope with the temporary existence of multiple addresses per host adds a complexity in management, but how much so is up to the end-users themselves. A user may choose to do direct transitions of some services (such as web servers) from one IP address to another, without going through a stage where the service is available on all addresses. While that is not strictly providing a fully seamless transition, it could significantly reduce the management complexity, without a significant impact on service, especially if the DNS updates are rapid.

It should also be noted that during a renumbering event, since the DNS resource record TTLs are significantly shorter, the primary DNS servers for the domains will receive significantly more queries, as resolvers should not cache the responses for so long, and will regularly check back with the master. The likelihood of this having any significant impact is, however, fairly minimal, at least in a typical small to medium site.

Section 3.1 of Baker [BAKER] is aptly titled "Find all the places", and serves as a gentle reminder to application developers that embedding addresses is bad at best. Where common UNIX tools such as "grep" allow administrators to crawl the file systems of servers for places where address information is hard-coded, the proliferation of technologies such as NetInfo and other directory- or hive-based configuration schemes makes the job of finding all the places that addresses are hard-coded intractable.

Beyond the call to arms for application and services developers made by Baker et al., and specific to the challenges of renumbering, the following security and policy-related services that initial research has flagged as particularly troublesome:

3.3.5.1. Packet filters, Firewalls and ACLs

Throughout the transition from the old address set to the new, all packet filters and firewalls will need to adapt to map policy to both prefixes (sets of addresses) - perhaps even selectively as the old addresses become deprecated. Whilst technologies such as Router Renumbering and Neighbour Discovery automate to a large extent the transition of router and node configurations, and dynamic DNS update for the re-mapping of resource records to reflect the new addresses [RFC2136], no such mechanism exists at present for mechanising the adaption of security policy.

Particularly troublesome policies to administer include egress filtering, where packet filters discard outbound packets that have source addresses that should not exist within the site, and filtering inbound site-local addresses in cases where two organisations are renumbering as a step toward merging their networks together (although the use of site-local addressing is now deprecated).

Where renumbering is due to a 'clean break' from previous connectivity provider, another consideration is for the ingress filtering performed by the provider. For instance, the new provider may refuse to receive into their routing topology those packets whose source address is under the old prefix, and likewise for the old provider and new prefix. Whilst it is not the business of the IETF to mandate business practice, it is likely that the provision of out-of-allocation prefix routing

as part of a multi-homing service contract would be a chargeable service and not one that an enterprise trying to make a clean break away would likely be willing to pay just for the duration of transition to their new prefix.

Beyond the immediate up-stream provider, there are other policy-based considerations to take into account when renumbering. Some rudimentary authenticated access mechanisms rely on access queries coming from a particular IP network, for example, and so those application service providers will need to update their access control lists. Likewise all the internal applications (possibly meant for 'internal' eyes only) will have to have their access controls updated to reflect the change. The use of symbolic access controls (i.e. DNS domain names) rather than embedded addresses may serve to mitigate much of the distributed administrative load here, at least if such symbols are re-resolved, especially during the mid-renumbering states where both sets of addresses are still live and valid.

3.3.5.2. Monitoring tools

Network monitoring and supervisory utilities such as RMON probes, etc., are often deployed to monitor network status based on IP traffic. During a renumbering episode, the addresses for which the probes should monitoring and the addresses of logging services to which the probes report (e.g. in the case of remote SNMP logging) need to be tracked.

"Helpdesk ops" service liveness monitoring software also poses a particular problem where liveness is determined, for example, by a null transaction (e.g. for POP3 mail server, authenticating and performing a NOOP) made against a named service instance, if the name is by IP then two instances of the liveness test will be required: one on the old address to cater for those remote parties that are not yet aware of the new address, and one test against the new.

As part of the renumbering process, it may be advantageous to deploy flow analysis tools that can be scripted to alert administrators on observation of particular traffic patterns, e.g. flows to a service under a deprecated prefix during transitions where both old and new prefixes are live and routed to the site concurrently. This can highlight, for example, mis-cached DNS resource records, sources of manually configured service location data, etc.

3.3.6. Considerations with a Dual-Stack Network

There are several issues to consider when renumbering a dual-stacked network. In the simplest case, the IPv4 addresses will be remaining the same while the IPv6 addresses are renumbered. This could, for example, be due to an upstream renumbering, a change of IPv6 transition method (such as a tunnel), or a topology change. In such a case, the IPv4 connectivity remains unchanged, and as such can be used as a fallback during the renumbering to assist with session continuity, DNS services, etc.

The other case is when the IPv4 network is being renumbered along with the IPv6 network. Again this could be due to an upstream change, a network reconfiguration, or because the two are inter-linked - such as with the 6to4 transition mechanism. In this case, it is unlikely that the existence of IPv4 on the network can be used for any advantage, and instead many of the same issues are likely to be found when renumbering the IPv4 network as for the IPv6 network, except for the fact that more of the renumbering must be manually configured, for example by reconfiguring the stateful IPv4 DHCP configuration, or even manually configuring IPv4 addresses.

A hybrid case is also possible, where IPv4 NAT is used on the internal network, but with globally routable IPv6 addresses. In this case, if both networks' external connectivity is being renumbered, the internal network will only see the effect of the IPv6 renumbering, while keeping the IPv4

addresses the same. The renumbering procedure will still have an impact on the IPv4 connectivity and its session survivability, however. It may also be possible that the site uses both global and ULA IPv6 prefixes, the ULA prefix being deployed to avoid impact to long-running IPv6 sessions.

3.3.7. Equipment administrative ownership

The question of who owns and administers (also, who is authorised to administer) the site's access router is an issue in some renumbering situations. In the enterprise scenarios, the liaison between the end users and remote administrators is likely to be relatively easy; this is less likely to be the case for a SOHO scenario. This is not likely to be a major issue, however, since SOHO renumbering is likely to only be required if the remote administrators deem it necessary, or if the end user is sufficiently technically competent and decides to renumber their own network.

3.4. Impact of Topology Design on Renumbering

This section looks at considerations regarding network design, such as network merging and NAT avoidance, that have an effect when undertaking a network renumbering event.

3.4.1. Merging networks

Renumbering of all or part of a network due to merging two or more smaller networks has many of the concerns already discussed, but it may not affect the whole network. For example, multiple disparate networks may be merged together as one entirely new subnet, and thus all hosts must be renumbered; but it is also possible that one of the networks in the merger retains its prefix, and the other network(s) merge with it.

When the networks merge, the router advertises itself, and the new prefix if appropriate, to the new hosts, and Duplicate Address Detection (DAD) [RFC2462] must be applied by the new hosts to ensure they are not taking addresses already assigned to the existing hosts. It is implementation-dependent, however, as to whether the DAD algorithm will be re-run on link-local addresses if the network configuration is changed, so there is the possibility of an address conflict. However, as is noted in RFC2462, DAD is not completely reliable, and as such it cannot be assumed that initially after a network merge all link-local addresses will be unique.

3.4.2. Fixed length subnets

The IAB/IESG recommendations for IPv6 address allocations [RFC3177] details some of the motivations behind the change in the addressing architecture of IPv6 since its inception, and asserts the current state of a 64-bit 'network' part (the prefix) and a 64-bit 'host' part (the interface identifier). Fixing the lower 64 bits to be exclusive of routing topology significantly reduces the administrative load associated with renumbering and re-subnetting as experienced with IPv4 networks previously, for example, to get better address utilisation efficiency as networks evolve and provider address allocations changed.

The recommendations also discuss what length of network prefix should be allocated to sites, typically provisioning for 16-bits of subnet space in which sites can build their topology. Having such a large address space for sites to divide up at their discretion alleviates many of the drivers for renumbering discussed during the old PIER working group's lifetime [RFC2071].

3.4.3. IPv6 NAT Avoidance

RFC2072, from the PIER WG, stated: "Network address translation (NAT) is a valuable technique for renumbering, or even for avoiding the need to renumber significant parts of an enterprise." That is, by 'hiding' the subnet topology and making independent of any connectivity provider the addressing model used within a site, NATs enable renumbering of entire networks because the only device that is renumbered when global addressing changes is the outside edge of the NAT devices.

However, NAT is strongly discouraged in IPv6, not least because it breaks end-to-end transparency (as described in [RFC2775]) and obscures identity - including the basis for permission, authorisation, verification and validation - and thus should not be considered as being available as a solution. A significant reason to deploy IPv6 is to simplify network and application operation by (IPv4) NAT removal, for example to provide true end-to-end connectivity, to make simple the gateway between site and Internet, to encourage 'considered' policy for secure access rather than rely on the (relatively) dangerous defence of 'hiding' behind a NAT. A more detailed discussion of the motivations for 'protecting' the network architecture from NATs can be found in [NAP].

3.5. Application and service-oriented Issues

In this section we highlight issues and common approaches to software development that 'disrupt' protocol layering to the extent that applications become aware of renumbering episodes, even if catastrophic and without knowing how to recover without failing.

3.5.1. Shims and sockets

Baker [BAKER] calls for application developers to consider the effects of renumbering whilst applications are 'live', particularly as regards caching the results of symbol resolution. Where applications maintain open connections to services over a sustained period of time (as opposed to the ephemeral nature of protocol interactions such as with HTTP), any change in either end's addressing may intrude on the application's execution - particularly if the change is abrupt or the session longer than the expiry and withdrawal time of the old addresses.

Various options may be available to minimise the risk of application disruption in this instance. A HIP-like 'shim' [HIPARCH], as is being developed as a candidate solution to the general multi-homing problem, removes the tight coupling between a connection and a service's topological location: as the renumbering event takes place, the locator is updated to reflect the new address topology, and the application remains blissfully unaware - a form of layer 3.5 mobility.

Alternatively, should the old address space be available such that a single (or subnet of) Mobile IPv6 Home Agents be deployed in the routing path of the to-be-otherwise-interrupted connection, then the endpoint being renumbered could utilise layer 3 mobility once the old prefix removed from its link, i.e. register with the Home Agent in the old prefix topology - presumably in the provider's network, formerly upstream from the site - and rely on Mobile IPv6 route optimisation to make good the additional overhead imposed by the reverse tunnelling to the new prefix.

Applications that employ SCTP as opposed TCP or UDP for communication avoid all of the issues highlighted in this sub-section due to the provision of dynamic endpoint reconfiguration in the protocol.

3.5.2. Explicitly named IP addresses

There are many places in the network where IP addresses are embedded as opposed to symbolic names, and finding them all to be updated during a renumbering episode is not a trivial task. This section details an evolving list of such places as surveyed as common.

Addresses may be hard-coded in software configuration files or services, in software source-code itself (which is particularly cumbersome if no source is available, e.g. a bespoke utility built to order), in firmware (for example, an access-controlling hardware dongle), or even in hardware, e.g. fixed by DIP switches.

A non-exhaustive list of instances of such addresses includes:

- Provider based prefix(es)
- Names resolved to IP addresses in firewall at startup time
- IP addresses in remote firewalls allowing access to remote services
- IP-based authentication in remote systems allowing access to online bibliographic resources
- IP address of both tunnel end points for IPv6 in IPv4 tunnel
- Hard-coded IP subnet configuration information
- IP addresses for static route targets
- Blocked SMTP server IP list (spam sources)
- Web *.htaccess* and remote access controls
- Apache *Listen* directive on given IP address
- Configured multicast rendezvous point
- TCP wrapper files
- Samba configuration files
- DNS *resolv.conf* on Unix
- Any network traffic monitoring tool
- NIS/ypbind via the hosts file
- Some interface configurations
- Unix portmap security masks
- NIS security masks
- PIM-SM Rendezvous Point address on multicast routers

Some hard-coded IP address information will be held in remote locations, e.g. remote firewalls, DNS glue, etc. adding to the complexity of the search for all instances of the old prefix. Should symbols be used rather than addresses, administrative ownership of DNS - with due consideration for the TTL of resource records - and other naming services ease this particularly problematic issue of data ownership and validity.

There are also cases when IP addresses are embedded into payload data, such as with UDP-based NFS mounts and FTP sessions.

3.5.3. API dilemma

There is an open question as to whether we need an extension to the sockets API that would allow applications resolving addresses to be able to determine the freshness of the resolved data. A straw poll of networking applications demonstrated that common programming practise is to 'resolve once, bind many' during the lifetime of an application, caching the initial lookup result and assuming that it is still valid throughout. Whilst this is a perfectly valid approach for short-lived applications, where the chance of renumbering - site or the single node - increases with regards the longevity of the application, the likelihood of the resolved data being intrusively inaccurate also increases.

Application programmers should therefore consider the possibility of network renumbering when writing socket software. The best behaviour is probably to freshly resolve for any socket binding, and let the resolver handle the caching, based on the DNS TTL. Only when there are a significant number of connections within a short timeframe should application-level caching be considered.

3.5.4. Server Sockets

Certain services create a server socket instance on which they intend to receive client connections throughout their execution lifetime, never re-binding that socket unless explicitly shut-down and restarted. An example would be a webserver, which may in fact bind to multiple different IP addresses to serve content for different domains where the particular business case is for customers to be allocated their 'own' IP address (e.g. for reverse DNS to reflect their branded domain name). Address space usage inefficiencies aside, the class of service that creates a server socket that persists on the initially-bound address is problematic during renumbering.

A typical work-around would be to schedule a restart of all such services having first identified whether they can operate on both address prefixes (to satisfy the middle states of Baker [BAKER]), or at least to schedule their migration to the new address configuration in light of the DNS name bindings (considering caches and TTL), and the nature of existing clients that may still be bound to the old service (consider graceful migration).

One possible solution worth considering, which is not yet implemented in APIs, would be to allow servers to bind to just the lowest 64 bits of an address, allowing the network identifier to change without the server knowing. This is a purely hypothetical solution, however, and has numerous issues, not least regarding requirements of some server software to know its current globally routable IP address.

4. SOHO Experiments (Southampton)

4.1. Basic SOHO Test (SoHo.0)

4.1.1. Purpose/Scenario

This scenario comprises a simple SoHo network for 'early adopters' of IPv6, with an ADSL Access Router to an IPv4-only upstream and using 6-to-4 and Brokered-IPv6-in-IPv4 tunnels for IPv6 connectivity for the local network.

The access router, 6-to-4 encapsulation node and tunnel endpoint is a Cisco 837 router, running Cisco IOS Version 12.3(7)XR2. The default valid and preferred lifetimes for router advertisements are 30 and 7 days respectively (2592000 and 604800 seconds).

4.1.2. Initial state

In the initial state, the router is allocated 2002: 5298:8249::/64 as a 6to4 network, with ::1 for its own address. The router is configured to explicitly route all 6to4 traffic toward a virtual tunnel interface, which encapsulates traffic in a 6to4 tunnel over the upstream's IPv4-only link toward the UoS 6to4 relay (NB: Not using IPv4 6to4 Anycast service in this experiment).

On the access router, the default IPv6 route is via the UoS 6to4 relay.

4.1.3. Preparation

The new (configured) IPv6-in-IPv4 tunnel interface created on the access router, and the ip-access-list rules mirrored for the new local address set in anticipation of the interface going live.

There is no DNS or stateful configuration service within the test environment.

4.1.4. Configuration / Activation

Adding prefix addresses to the configured tunnel interface, the LAN-facing Ethernet interface, and local static routing were applied.

The ensuing Router Advertisement was picked up by all of the clients (Apple Mac OS/X 10.3.7, Microsoft XP SP2, RedHat Fedora Core 3 Linux 2.6.9-1.667) with correct Valid and Preferred lifetimes.

It was noted that the nature of the IOS CLI meant that the new prefix was activated whilst configuration was on-going (c.f. SoHo.1 experiment below). Administrators performing operational renumbering on larger networks should consider not editing the running configuration on a router in a production environment, but rather a file copy that can be applied on router reload (at the trade-off of downtime and live connection-tracking state loss whilst configuration reapplies).

At this stage, the experiment was terminated due to issues with local clients and open questions regarding the ability for administrators to manually step-down prefix validity and preferred lifetimes.

4.1.5. Summary

Connection survivability across the access router from external to internal networks was not tested in this experiment. Also, the scheduled step-down of Valid and Preferred lifetimes were not tested at this time.

This experiment is to be repeated (in full, with deprecation and removal) once UoS receive native IPv6 ADSL (expected late-January 2005), so as to renumber from tunnelled connectivity to native.

4.2. SoHo.1

4.2.1. Purpose/Scenario

This scenario involves a typical 'advanced' SoHo network, where IPv6 transition mechanisms are in use. The purpose of this renumbering is the transition from 6to4 to an IPv6-in-IPv4 tunnel.

The Internet connection router and tunnel endpoint is a Linux machine, running kernel 2.6.7 and Debian. Client machines on the network in this experiment were Windows XP SP1, Debian Linux (kernel 2.4.27), and FreeBSD 4.9. radvd is used on the router for stateless address auto-configuration. The default valid and preferred lifetimes are 30 days and 7 days respectively.

4.2.2. Initial state

In the initial state, the router is allocated 2002:abcd:efab:/48 as its 6to4 address block, and takes 2002:abcd:efab:0::1 for its own address and the associated /64 as the subnet for the main Ethernet interface.

For testing purposes, ssh sessions to and from external hosts, as well as communication between the internal hosts, were established.

4.2.3. Preparation

The tunnel endpoints were established at the tunnel server and the local router. No DHCPv6 or special DNS addresses were in use, so no reconfiguration was required there.

The ip6tables firewall on the router was reconfigured to include duplicates of any rules that explicitly mentioned addresses, changed to refer to the new prefix.

4.2.4. Configuration

At this stage, the new prefix addresses were added to the tunnel interface, and also to the Ethernet interface, along with a route.

The IPv6 default route of 2000::/3, metric 256, was added to the router, pointing at this new tunnel interface.

At this stage, all machines were still configured using the 2002: 6to4 address, and as such all the old connections continued to work, and new connections would still come from these addresses.

4.2.5. Activation

The new prefix was configured in radvd.conf for the Ethernet interface, containing the same defaults as for the old 6to4 prefix.

Upon restarting radvd, all machines on the network correctly picked up their new addresses. Internal ssh sessions between these addresses worked fine. All of the ssh sessions that had been established for testing purposes remained alive and working. The machines were also correctly listening for new ssh requests on their new addresses.

New connections established from all machines on the network now came from the new prefix, despite them being given equal validity, preferred lifetime, and the default routes at the router having equal metrics.

4.2.6. Deprecation

The metric of the old default route was decreased, and the old prefix was marked as deprecated in radvd (valid lifetime of 3600 seconds, preferred of 0). This was correctly noted in all three operating systems, although the valid lifetime was increased to 7200 seconds (2 hours), because of the recommendations in RFC2462. The lifetime of the route is marked as 3600 seconds (under Linux, where this information is visible), but does not decrement consistently.

Incoming ssh sessions to both addresses were still correctly working at this stage.

Outgoing connections continue to come from the correct (new) addresses. All established sessions continue to work correctly.

All operating systems tested were behaving in the same fashion at this stage.

4.2.7. Removal

After all hosts have picked up the new router advertisements with the prefix deprecation and 2 hour lifetimes, the old prefix was completely removed from radvd.conf. It now became up to the operating systems to handle the final removal of the addresses and routes from the interfaces.

Initially, all the machines still have the old prefix, and it remains in existence for the whole of the valid lifetime. They will accept incoming connections on these deprecated addresses, but will not use them for source.

Finally, once the validity of the old-prefixed addresses drops to zero, it is removed from the interfaces of all the operating systems, and then any existing sessions using those addresses were dropped.

At this point, it is safe to remove any remaining routing for the old prefix from the router, without further disrupting connectivity.

In one renumbering experiment, the Linux machine, and possibly others, kept their old prefix addresses for considerably longer than was claimed with the valid lifetime. The counter did not always decrement as expected, and the two hour delay until invalidation in fact took over eight hours. This behaviour has not yet been possible to replicate, however.

4.2.8. Stable state

Once the renumbering has been complete, and a single prefix is again in use, the machines on the network behave in a stable fashion.

4.2.9. Summary

Renumbering in a SoHo environment has proven to be a relatively seamless task. Issues were raised in the first experiments undertaken, with the difficulties in using a single /64 for routing. This was resolved by using a /60, with a /64 then split into subnets for the main internal Ethernet network.

In the documented scenario, the introduction of a new prefix onto the local network caused no problems. The new (tunnelled) prefix, rather than the old (6to4) prefix, appeared to be preferred in all new connection situations, even before deprecation. Sessions that had been initiated using the old prefix remained active until the prefix was declared invalid and removed from the hosts. After the renumbering procedure had been completed, there was stable use of the new prefix.

4.3. Utility for generating Router Advertisements (sendra)

Initial investigation into stack behaviour across a number of platforms demonstrated that it was problematic to precisely control what Router Advertisement packets nodes would receive so as to observe specific behaviour and match to expectations.

A small utility application has been developed to generate advertisements with a very fine level of control.

```
sendra -i <interface>
    [-d <destination (default ff02::1)>]
    [-h <cur hop limit (default 0, i.e. unspecified)>]
    [-M (set managed bit)]
    [-O (set other bit)]
```

```
    [-l <router lifetime (default 0, i.e. not default router)>]
    [-r <reachable time (default 0, i.e. unspecified)>]
    [-t <retrans timer (default 0, i.e. unspecified)>]
RA options, non are mandatory, and will be in listed order
    [-s <source link-layer>]
    [-m <MTU>]
    [-p <prefix information (may be repeated)>]
The prefix information has the following format
    <address>/<len>,<onlink {0,1})>,<auto {0,1})>,<valid
time>,<preferred time>
See RFC 2461 for further explanation
```

The purpose of the utility was to allow testing of everything in RFC 2461, with reasonable default configuration values, mirroring typical router behaviour.

The utility only sends one single advert as specified, requiring scripting or other invocations to achieve appropriate ramped-delays between RAs, etc.

An example sequence of invocations that demonstrate the utility in use would be as follows.

```
sendra -i eth0 -s -p 2001:800:1::/64,1,1,3600,14400
sleep 60
sendra -i eth0 -s -p 2001:800:1::/64,1,1,3600,14400 -p
2001:700:1::/64,1,1,3600,14400
sleep 60
sendra -i eth0 -s -p 2001:800:1::/64,1,1,0,3600 -p
2001:700:1::/64,1,1,3600,14400
sleep 60
sendra -i eth0 -s -p 2001:700:1::/64,1,1,3600,14400
```

5. SOHO Renumbering Experiment with a Tunnel Broker (JOIN)

In the past a lot of people owned only one PC and connected to the internet over dial in modem or ISDN connections. Recently more and more broad band connections like cable modem and DSL are used to connect not only one PC but several ones of a small business or several family members. With IPv6 a dial in account should get a /64 prefix if only one subnet is needed, i.e. in a home environment, or a /48 prefix if more than one subnet is needed, i.e. in a small business [RFC3177].

With a lot of different ISPs and internet costs constantly dropping it is not uncommon that dial in customers switch often between the cheapest ISPs and are 24 hours a day online. In most cases these customers use only client software like web browsers, email, ftp or ssh clients which are in general not as important as business services. When switching an ISP it is unavoidable that a

download gets cancelled or an SSH session stalls. The other point is what happens to server applications like web server or ssh daemons after changing addresses? If the socket API is correctly used changing the IP address should not hinder future connection attempts. This is the main point to be checked.

5.1. The Tunnel Broker

Since we do not have native IPv6 support in the whole university we use a tunnel broker to offer IPv6 connectivity for single hosts resp. subnets from students and employees. The tunnel broker utilizes OpenVPN[2] to create a secure VPN tunnel to one of JOIN's native IPv6 connected computers which is acting as a server. The client gets a /64 subnet and a global unique IPv6 address out of the University of Muenster's address space (2001:638:500::/48). All IPv6 traffic gets routed over the tunnel. We use a client computer resp. several computers in a subnet connected by the tunnel broker to the IPv6 network exemplarily for a home network connected per dial in.

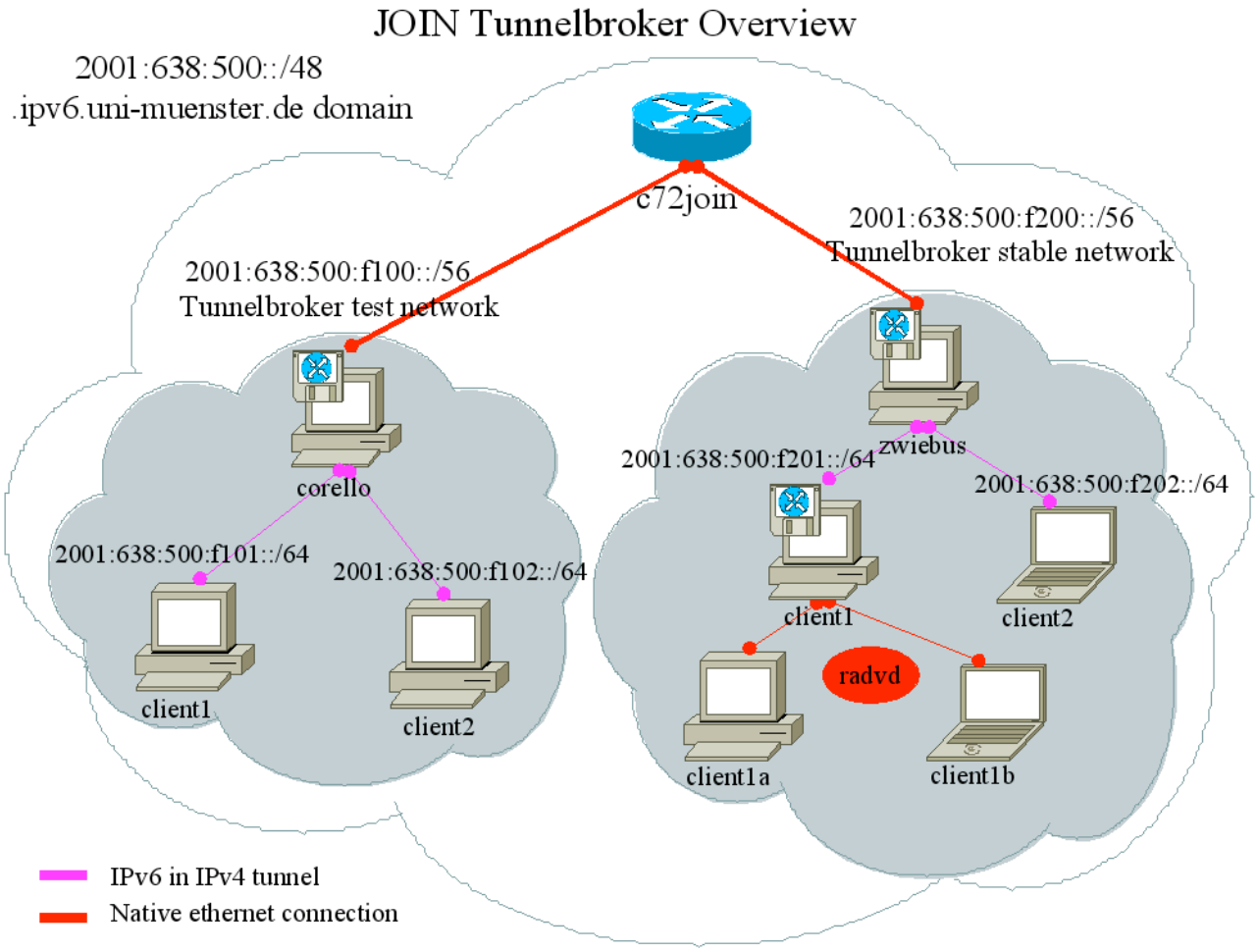

Figure 5-1: JOIN Tunnel broker

[2] http://openvpn.sourceforge.net

5.2. Renumbering procedure

There are two possibilities when renumbering a home network can get necessary. First one is if the customer finishes a session with its ISP and starts a new one. Normally when changing the ISP a short network outage cannot be avoided. It is not common to be multihomed using two different ISPs. The second possibility is when the ISP itself renumbers or sends another prefix to its customer.

With the tunnel broker we are able to simulate these cases. It is possible to open one or multiple tunnel connections and get one or multiple /64 prefixes for a single computer or subnet. The computer with the tunnel broker client advertises the /64 prefix(es) via radvd[3] in its subnet.

With respect to Baker's renumbering procedures [BAKER] for enterprise networks there are only three steps left over in SOHO renumbering. These are the stable old prefix (1), two prefixes (5) and the new prefix (8). Network management options and name service configuration do not apply in a SOHO network.

5.3. Discovered Problems

We have been busy with the setup of the environment and have so far tested only some server software products that are widely used in SOHO networks. These are OpenSSH server, ProFTP server, Apache HTTP server, a NTP server and an OpenMCU server. All servers responded on both IPv6 addresses during transition without problems. The results were that protocols which open a new connection on every request like FTP, HTTP or NTP and do not open long sessions are not much affected by the renumbering. On the other side do long SSH sessions, OpenMCU sessions or large file downloads stall and fail as expected when removing the old address/prefix. This cannot be avoided.

5.4. Special tests

The results of the up to now done tests could be expected. So what special tests have to be done?

- Check handling of valid/preferred life times: i.e. linux 2.6.8.1 counted preferred lifetimes below zero. Is this right?
- Selection of source address (RFC3484): Are the rules of the RFC already implemented? Are the specified rules sufficient for real world scenario?
- Check TCP listening/connected state: Do applications listen on ANY address or only specified addresses. Is it a configuration issue?

5.5. Configuration Information

Here we highlight prefix-specific information in configuration files.

5.5.1. OpenVPN server configuration

zwiebus$ cat /etc/openvpn/server-conf/openvpn1.conf

[3] http://v6web.litech.org/radvd/

```
# Main configuration, start as daemon or from (x)inetd, tun device and port
inetd server-openvpn1
dev tun
tun-ipv6
port 5011
up /etc/openvpn/server-conf/openvpn1.up

# Security options
# Switch to user nobody
user nobody

# Don't encrypt data channel for better performance
cipher none

# TLS server setup
tls-server
dh /etc/openvpn/ssl/dh1024.pem
ca /etc/openvpn/ssl/zwiebus-ca.crt
cert /etc/openvpn/ssl/servers/openvpn1.crt
key /etc/openvpn/ssl/servers/openvpn1.key
tls-verify /etc/openvpn/server-conf/openvpn1.tls-verify

# Debugging
log-append /etc/openvpn/server-logs/openvpn1.log
persist-tun
ping 15
ping-restart 45
ping-timer-rem
persist-key
verb 4

# Exit after 10 min of idle time
inactive 600
```

zwiebus$ cat /etc/openvpn/server-conf/openvpn1.up

```
#!/bin/bash

INTERFACE=$1; shift;
TUN_MTU=$1;   shift;
UDP_MTU=$1;   shift;
LOCAL_IP=$1;  shift;
REMOTE_IP=$1; shift;
MODUS=$1;     shift;

ip link set ${INTERFACE} up
```

```
ip link set mtu ${TUN_MTU} dev ${INTERFACE}

ip -6 addr add 2001:638:500:f200::0002:1/112 dev ${INTERFACE}
ip -6 addr add fe80::2:1/64 dev ${INTERFACE}
ip -6 route add 2001:638:500:f202::/64 dev ${INTERFACE}

exit 0
```

5.5.2. OpenVPN client configuration

flipper$ cat /etc/openvpn/openvpn1.conf

```
# Main configuration, start as daemon
daemon client-openvpn1
dev tun
tun-ipv6
port 5011
up /etc/openvpn/openvpn1.up

# Target host, OpenVPN server
remote zwiebus.uni-muenster.de

# Security options
# Don't encrypt data channel for better performance
cipher none

# TLS client setup
tls-client
ca /etc/openvpn/zwiebus-ca.crt
cert /etc/openvpn/openvpn1.crt
key /etc/openvpn/openvpn1.key
tls-verify /etc/openvpn/zwiebus.tls-verify

# Debugging
# mtu-test
# tun-mtu 1500
# tun-mtu-extra 32
log-append /etc/openvpn/client-logs/openvpn1.log
persist-tun
ping 15
ping-restart 45
ping-timer-rem
persist-key
verb 4
```

flipper$ cat /etc/openvpn/openvpn1.up

```
#!/bin/bash

INTERFACE=$1; shift;
TUN_MTU=$1;   shift;
UDP_MTU=$1;   shift;
LOCAL_IP=$1;  shift;
REMOTE_IP=$1; shift;
MODUS=$1;     shift;

ip link set ${INTERFACE} up
ip link set mtu ${TUN_MTU} dev ${INTERFACE}

ip -6 addr add 2001:638:500:f200::2:2/112 dev ${INTERFACE}
ip -6 addr add fe80::2:2/64 dev ${INTERFACE}
ip -6 route add default dev ${INTERFACE} metric 1

sysctl -w net.ipv6.conf.all.forwarding=1

ip -6 addr show dev eth0 | grep 2001:638:500:f202::1/64 >/dev/null 2>&1 || ip -6 addr add 2001:638:500:f202::1/64 dev eth0
```

5.5.3. radvd configuration

flipper$ cat /etc/radvd/radvd.conf

```
interface eth0
{
 # Needs to be on to enable advertisement on this interface
 AdvSendAdvert on;

 # These settings cause advertisements to be sent every 3-10 seconds.  This
 # range is good for 6to4 with a dynamic IPv4 address, but can be greatly
 # increased when not using 6to4 prefixes.
 MinRtrAdvInterval 3;
 MaxRtrAdvInterval 10;

 # Disable Mobile IPv6 support
 AdvHomeAgentFlag off;

 # First prefix from tunnelbroker
 prefix 2001:638:500:f202::/64
 {
  AdvOnLink on;
  AdvAutonomous on;
  # AdvRouterAddr off;
```

```
    AdvValidLifetime 300;
    AdvPreferredLifetime 60;
    # AdvDefaultLifetime 900;
  };

  # Second prefix from tunnelbroker
#  prefix 2001:638:500:f203::/64
#  {
#    AdvOnLink on;
#    AdvAutonomous on;
#    # AdvRouterAddr off;
#    AdvValidLifetime 300;
#    AdvPreferredLifetime 60;
#    # AdvDefaultLifetime 900;
#  };
};
```

5.5.4. Client network configuration

flipper$ ip -6 addr

```
1: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qlen 1000
    inet6 2001:638:500:f203::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 2001:638:500:f202::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::2d0:59ff:fe2a:eddb/64 scope link
       valid_lft forever preferred_lft forever
2: lo: <LOOPBACK,UP> mtu 16436
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
7: tun0: <POINTOPOINT,MULTICAST,NOARP,UP> mtu 1500 qlen 100
    inet6 fe80::2:2/64 scope link
       valid_lft forever preferred_lft forever
    inet6 2001:638:500:f200::2:2/112 scope global
       valid_lft forever preferred_lft forever
8: tun1: <POINTOPOINT,MULTICAST,NOARP,UP> mtu 1500 qlen 100
    inet6 fe80::3:2/64 scope link
       valid_lft forever preferred_lft forever
    inet6 2001:638:500:f200::3:2/112 scope global
       valid_lft forever preferred_lft forever
```

flipper$ ip -6 route

```
2001:638:500:f200::2:0/112 dev tun0  metric 256  mtu 1500 advmss 1440 metric10 64
2001:638:500:f200::3:0/112 dev tun1  metric 256  mtu 1500 advmss 1440 metric10 64
```

```
2001:638:500:f202::/64 dev eth0  metric 256  mtu 1500 advmss 1440 metric10 64
2001:638:500:f203::/64 dev eth0  metric 256  mtu 1500 advmss 1440 metric10 64
fe80::/64 dev eth0  metric 256  mtu 1500 advmss 1440 metric10 64
fe80::/64 dev tun0  metric 256  mtu 1500 advmss 1440 metric10 64
fe80::/64 dev tun1  metric 256  mtu 1500 advmss 1440 metric10 64
ff00::/8 dev eth0  metric 256  mtu 1500 advmss 1440 metric10 1
ff00::/8 dev tun0  metric 256  mtu 1500 advmss 1440 metric10 1
ff00::/8 dev tun1  metric 256  mtu 1500 advmss 1440 metric10 1
default dev tun0  metric 1  mtu 1500 advmss 1440 metric10 64
default dev tun1  metric 1  mtu 1500 advmss 1440 metric10 64

alaska$ ip -6 addr
1: lo: <LOOPBACK,UP> mtu 16436
   inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
4: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qlen 1000
   inet6 2001:638:500:f202:201:2ff:fedd:5056/64 scope global dynamic
      valid_lft 300sec preferred_lft 60sec
   inet6 2001:638:500:f203:201:2ff:fedd:5056/64 scope global dynamic
      valid_lft 300sec preferred_lft 60sec
   inet6 fe80::201:2ff:fedd:5056/64 scope link
      valid_lft forever preferred_lft forever
```

6. Issues in 'Two Prefixes on a Link' in IPv6 (JOIN)

One of the most profound differences between IPv6 and IPv4 is the fact that with IPv6 an interface is no longer limited to a have only a single IP address assigned to it. While there is a strict one-to-one mapping between an IPv4 address and an interface (be it a virtual or physical interface) IPv6 offers the possibility to assign multiple IPv6 addresses to any interface. In fact every IPv6 interface, which is globally connected already bears at least four addresses from different scopes.

IPv6 addresses might either vary in the latter 64 bits (the host-id) or the first 64 bit (the prefix). If an interface is configured with two address where only the host IDs differ this hardly poses a problem for the host. If there are however two addresses with different prefixes this can affect forwarding and routing of packets that are originated from or destined for this interfaces depending on which of the addresses was selected as the source address for outgoing packets. This situation also implies that the two prefixes are *on the link*. A prefix is said to be *on-link* when the two following statements apply:

- A router has a static route for that prefix configured on an interface connected to the link.
- The network routing is configured such that packets destined for addresses from that prefix, which originate anywhere within the prefix's scope (global or site-local), reach that router through the connected network.

Among other scenarios a renumbering event will lead to a period of time, where there are at least two prefixes (one old, one new) valid on a link. In this section we want to outline and explain these effects that may result from this situation. Additionally we will describe possible security implications as well as provide solutions on how to avoid known/expected problems.

This section is called 'two prefixes on a single link' because it is sufficient to discuss this case for two different prefixes. Three or more prefixes will show similar effects and will not increase complexity. This section describes a scenario where interfaces are statically connected to one link and where there is more than one unicast prefix of greater than link-local scope valid on this link. Specifically we are not talking about mobility or Mobile IP were renumbering is very much an issue, too, but which is out of scope for this document. We may however discuss solutions for this case in the mobile world to see how far they may be applicable to our scenarios.

6.1. Theory

6.1.1. Interface Address Types and Prefix Scopes

An IPv6 interface is always configured with a variety of IPv6 addresses. For instance, at any time it bears at least one link-local and one multicast address without which it cannot work. However in the context of this deliverable only some of the various address types are of interest for us.

6.1.1.1. The Link-Local Scope

When an IPv6 interface is initialized, it immediately begins with the process of stateless address autoconfiguration to assign a link-local address to itself (e.g. fe80::204:75ff:fe82:e0ca). It uses this address to communicate with other interfaces on the link. Note that this process is mandatory even if stateful address autoconfiguration (DHCPv6 [RFC3315]) is used to later configure global or site scoped addresses. A link-local address is a vital part of an IPv6 interface as it is e. g. used to contact the router (sending Router Solicitations [RFC2461]), to discover its environment (receiving a global or site-scoped prefix via Router Advertisements or alternatively to talk to a DHCPv6 server/relay.

For our discussion link-local addresses are of no special interest, as they are chosen and assigned by the interface itself and are only meant and used for communication on the link. While it is possible to have multiple link-local addresses on an interface, doing so neither makes sense nor harm.

6.1.1.2. The Global Scope

When a node wants to communicate with other nodes on the Internet worldwide it needs to use an address or rather prefix, which has global scope. These type of prefixes are up to 64 bits long and begin with the first three bits set to 001 (e.g. like in 2001:638:500:200::/64 or 2002:80b0:bf03:1::/64). These prefixes were formerly named globally aggregatable (RFC 2374 and RFC 3587) because aggregated shorter prefixes of this form (up to 32 bits) constitute the global routing table. A node can be located from anywhere on the Internet, based on the first 64 bits of it's address alone, which is why IP packets can be forwarded on the correct way to their destination address.

Prefixes with global-scope are usually advertised on links in the form of RAs or via a stateful mechanism (most likely DHCPv6).

6.1.1.3. The Site-Local Scope

In contrast to prefixes with global scope, site-local prefixes are only valid within a site. These types of prefixes begin with the first 7 bits set to 1111110 or 1111111 (fc00::/7 or fd00::/7, see [ULA]

and [C-ULA]), e. g. fc11:2233:4455:1000::/64). They are not valid beyond the site's border router, meaning that they are not present in the global routing table and that a node outside the site is not able to locate and therefore not able to communicate with this interface address.

Just like global-scope prefixes, site-local prefixes can be advertised in RAs or via a stateful mechanism on a link.

6.1.1.4. Multicast Addresses

Multicast addresses are automatically chosen and configured by the interface or the node itself, depending on which multicast group it wants to join. Multicast addresses always begin with the first 8 bits set to 1 (ff00::/8) [RFC3513], e.g. ff02:2 and may have a different scopes, indicated by the forth nibble of the address. So e.g. ff05::/16 means site-local scope and ff0e::/16 means global scope. There are several scope levels in between these two scopes [RFC3513]. Aside from link-local multicast an interface has to send out a group join message (usually through MLD (see RFC 3810) to receive multicast traffic. After this message was sent the links router keeps track of which interfaces belong to which multicast group and forwards packets accordingly.

Link scoped multicast (ff02::/16) is a special case, where no MLD messages are necessary. These multicast addresses are only valid on the link and do not need to be announced beyond a router. For instance, every node must assign the *all node(s) multicast address* (ff02::1) and the correct *solicited multicast address* [RFC3513] to its interface.

Just like link-local addresses, multicast addresses in general are unrelated to the issues we want to address in this document. Multicast addresses are either only valid on a link or used to reflect to which multicast group an interface belongs. Therefore in this context the term scope does not include the different multicast scopes.

6.1.2. How does a prefix get to be on-link?

Two prefixes on a link only have an effect on network operation when there are in fact also hosts/nodes on the link having configures addresses from both prefixes. Address configuration happens either automatically by the stateless (SLAAC) or stateful (DHCPv6) approach or manually.

6.1.2.1. Stateless Address Autoconfiguration

With IPv6 there are two kinds of (address) autoconfiguration a stateless and a stateful approach. The latter involves the use of DHCPv6, which is briefly described below. Stateless address autoconfiguration (SLAAC) is totally new with no corresponding functionality in IPv4.

IPv6 Stateless Address Autoconfiguration (SLAAC) is described in [RFC2462]. It needs neither special client software nor servers. The functionality to use it is included in any standard compliant IPv6 implementation on hosts and routers.

For IPv6 hosts no manual configuration or setup is required to use the procedure as SLAAC is the default and works out of the box.

SLAAC enables a host to generate addresses for all its interfaces. To do this the host only needs some information about itself (in most cases its MAC address(es)) and some information sent by a router on the link as special packets, the so called router advertisements (RAs). RAs are designed specifically for one link and include information about prefixes which are on-link as well as some flags concerning whether or not hosts may use certain prefixes to configure themselves with addresses from that prefix and if the router itself may be configured as a default router.

Upon initialisation of the IPv6 protocol a hosts immediately starts the SLAAC process by first configuring link local addresses of all network interfaces. Link local addresses are composed by

adding a host identifier created from the MAC-address to the link local prefix fe80::/64. After a link local address is configured (and confirmed to be unique on the link) the hosts sends out a Router Solicitation (RS) message to find out if there are any routers present on the link. Upon reception of an RS a router may send an RA to the hosts having sent the RS using its link local address. If stateful autoconfiguration is to be used by the clients to configure addresses and default routes, a router may not send any RAs, even if it is a default router for the link.

If RAs are to be sent to the link by the router they also contain a flag, which may tell hosts to use DHCPv6 for address autoconfiguration. A host may use any combination of autoconfiguration including configuring addresses both by the stateless and the stateful procedure.

Router advertisements further include zero or more prefix fields. Each prefix may be individually set for use in address autoconfiguration. As stated previously hosts use these to generate the first 64 bits of their IPv6 addresses. The last 64 bits are by default generated the same way as for the link local addresses.

6.1.2.2. DHCPv6

DHCPv6 (Dynamic Host Configuration Protocol for IPv6) includes a stateful method to let hosts automatically configure site-local or global scope IPv6 addresses. It also offers the possibility to autoconfigure a multitude of further hosts settings (DNS, NTP, SIP-Servers, etc.), which are of no particular relevance during the renumbering process.

6.1.2.3. Manual Configuration

With IPv6 addresses may of course also be configured manually, overriding any of the both methods mentioned above.

6.1.3. When are there 'Two Prefixes on a Single Link'?

There are a number of scenarios where one might see two prefixes on a single link.

6.1.3.1. Multihoming

In the context of this document we talk about multihoming as a case where a site has two ISPs providing (global) connectivity in parallel. Each provider will assign the site with a prefix from its own address block. Every host within the site will only then be able to make full use of the redundancy made available by the multiple providers, when the whole site is configured with at least one address from each provider. This means, that there must be prefixes from all providers on every link.

Traffic originating outside the site destined for an address from provider A will reach the site through provider A, traffic destined to an address from provider B will reach the site through provider B. Additionally a provider may employ ingress filtering on the interface to the customer and not forward any traffic with different source addresses than from its own prefix. Both of these implications may already lead to a multitude of problems, which also apply to renumbering.

6.1.3.2. Renumbering

During the process of renumbering both the old prefix and the new prefix will be used in parallel on every link for a while effectively leading to it being multihomed for that time period. During this time most of the problems concerning regular multihoming also apply.

6.1.3.3. Site-Local (ULA) and Global Scope

Sites, which are not always connected to the Internet, may use site-local addresses [ULA] [C-ULA] for normal intra-site communication. When they connect to the Internet they may be assigned an additional global prefix and for the time they stay connected have two prefixes configured.

6.2. Problems

The following issues have been identified.

6.2.1. Source Address Selection

When a node establishes a connection over the Internet, it does not only need a destination address, but also a source address to establish the communication. Usually a node uses the IP address of the outgoing interface to put in the Source Address field in the IP packet. With multiple addresses available however a node has to choose, which address it wants to use as source address. Depending on this choice, several problems may occur.

6.2.1.1. Asymmetric routing

If the node can choose between addresses with prefixes from different providers, the packet might leave through ISP A, while the source address is picked from ISP B's prefix. The answer to that packet will return through ISP B and the route is asymmetric. This fact does not cause a direct problem, but it is undesirable for network management and debugging reasons. It might also lead to packet loss, if special filters are applied.

6.2.1.2. RPF filtering

It might happen, that a packet with a source address from ISP A's prefix leaves the site through ISP B. Sometimes ISPs add ingress filters (strict RPF checks (RFC 3704)) to their customers to prevent IP spoofing (RFC 2827). If this is the case for ISP B in this scenario, the IPv6 packet with the source address from ISP A will be dropped.

6.2.1.3. Wrong Scope

With IPv6 even unicast addresses have a scope, namely link-local, site-local and global scope. Link-local addresses are only valid on a link, while site-local addresses are filtered at the border of a site.

If an interface is configured with multiple addresses with different scopes, especially addresses from the global scope and addresses from only site-local scope, and chooses a site-local address as source address when talking to a node outside the site's scope, the communication will fail as the returning packets cannot reach the non-global address.

6.2.2. SLAAC, Address Lifetimes and RA Timers

6.2.2.1. Two routers on a Single Link, Minimum RA Interval

As per [RFC 2461], Router Advertisements are sent out with a minimum interval of 3 seconds (MIN_DELAY_BETWEEN_RAS). This interval exists to rate limit the traffic on a link.

This resulted in some implementations also dropping RAs they receive, when they reach the node faster than this minimum delay of three seconds. This fact may lead to problems with scenarios where there is more than one router on the link, each of which sending out router advertisements in different intervals. The intervals may inadvertently reach a phase where one or many router advertisements reach the nodes with less than a 3 seconds time interval resulting in one or more of

them being ignored by the hosts and default route and address lifetimes not being reset. If these lifetimes are short -- as might be the case during a renumbering event -- this may lead to premature timeouts of addresses or default routes and thus broken connections.

6.2.2.2. Lifetime Timeouts

Some implementations do not react properly to lifetime timeouts. While most of the implementations do respect the values set by an RA and decrease the valid lifetime as intended, some do not remove the address or route from the stack after the lifetime times out. This is especially malicious in case of the removal of a default route. We have witnessed at least one implementation, where instead of deleting the route from the valid default route table, the counters jumped to over 2 million seconds again and counted down from there.

Some older linux kernels also had problems with addresses in that regard and continued to use addresses on their stacks after the valid lifetimes had timed out, because after reaching zero they counted down to negative values.

6.2.3. Misconfiguration

6.2.3.1. Announcing a Non-Valid Prefix

Sometimes it is easy to break communication on a link by advertising a prefix by mistake or by announcing wrong lifetime values.

For example there are some Linux distributions that activate a service that sends out RAs the moment it is installed (e.g. radvd). The service then uses a default configuration and instantly sends router advertisements with a site-local prefix to the link or at least offers itself as a default router. In most cases this is activated by mistake and the client node does not offer any connectivity outside the link at all much less a route to the global Internet. In this case all pre-existing communication on the link may break.

6.2.3.2. Wrong Timers

In case of a renumbering event, timers have to be adjusted to prepare a prefix and maybe even default route to time out faster than usual.

RAs have to be sent in a fixed interval. The timers used for valid lifetimes of the prefix and the default route are reset whenever an RA reaches the node. Due to this the values for the valid lifetime should always be significant larger than the maximum RA interval.

These "overlaps" could get broken when the lifetime values are set to a low value as a preparation for a renumbering event. This is especially true if the prefix information is sent by a different node than the default route information, or in general, if the new prefix and default route is handled by a second router with different timers.

6.2.4. Client behaviour with Default Route/Address Selection

When multiple prefixes are used on a link they are either send by a single router in a list or are send by different entities, like two different routers sending out RAs, or a configuration where one prefix is send via RA and others via DHCPv6.

Implementations tend to react differently on which prefix they use primarily once they received it. Even when proper source address selection is implemented, there are prefixes that are equally weighted and the nodes can decide on their own, which they want to use as source address.

Comparing the implementations, it is e.g. inconsistent if they use the first or the last prefix that came with in a list in a RA. An implementation might choose to use always the last prefix it received. This could get even worse, when two routers send different RAs and the node always flip flops between both prefixes.

Overall, this might lead to inconsistent behaviour of different nodes with different implementations and could hinder a renumbering process.

7. Renumbering an IPv6 Backbone Network (JOIN)

In this section we describe the process that JOIN went through in renumbering its backbone network, the 6WiN.

7.1. The 6Win Scenario

6WiN is the native German IPv6 test network that has been established in parallel to the G-WiN (the German high-speed NREN) by the JOIN-Team in cooperation with T-Systems Nova Berkom. It is an IPv6-only core-to-edge network that runs on dedicated hardware (Cisco 7206, 34mbit E3) independently from the IPv4-only G-WiN. The 6WiN is still maintained by JOIN, the IPv6 working group at the University of Münster.

The prefix 2001:0638::/32 has been assigned by RIPE to DFN-Verein from which 2001:0638:0::/48 has been reserved for the 6WiN backbone. The DFN-Verein is a non-profit association of the research, development and education sector in Germany to promote computer-based communication and information services. Founded in 1984 as an institution comprising universities and technical colleges, non-university research institutions and private sector research bodies, the DFN association currently has 370 institutional members. The gigabit NREN G-WiN is the German pendant piece to the North American Internet2 initiative ABILENE and one of the worlds most advanced communication networks.

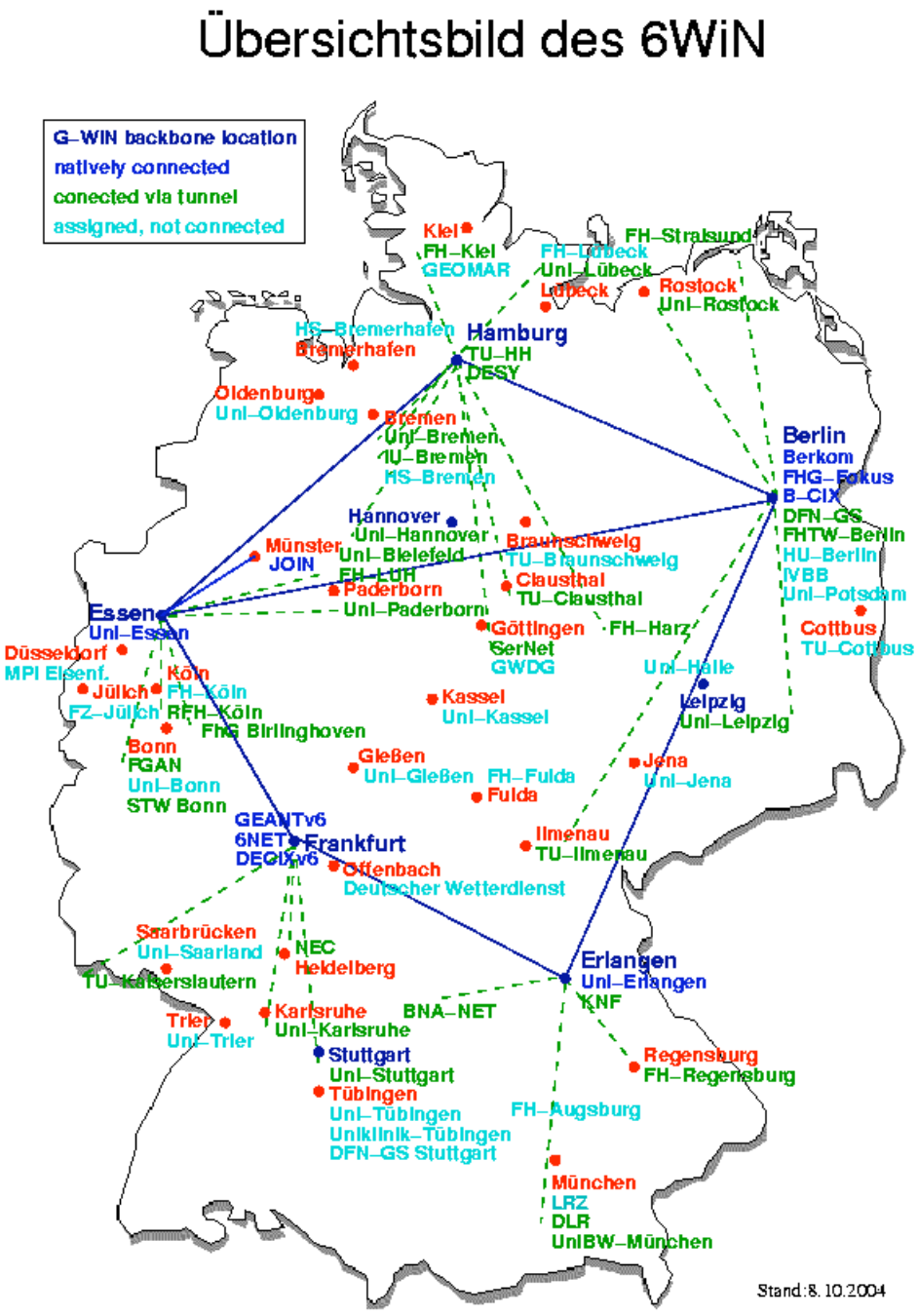

Figure 7-1: The 6WiN network

The 6WiN backbone consists of 5 pops, which were chosen to be near primary G-WiN locations and evenly distributed in Germany.

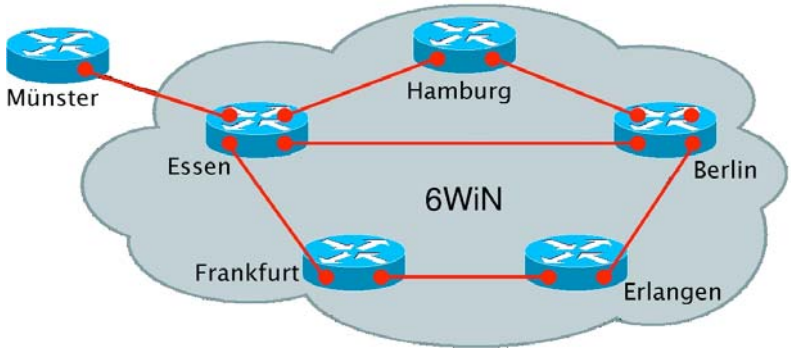

Figure 7-2: 6WiN topology overview

The links are as follows:

- `Frankfurt: nr-fra1.6win.de (2001:638:0:300::1)`
- `Essen: nr-ess1.6win.de (2001:638:0:500::1)`
- `Hamburg: nr-ham1.6win.de (2001:638:0:600::1)`
- `Berlin: nr-ber1.6win.de (2001:638:0:800::1)`
- `Erlangen: nr-erl1.6win.de (2001:638:0:a00::1)`
- `Münster:` (As of 1.7.2004 Münster is only connected via GRE-tunnel to 6WiN while it was formerly connected by a 34mbit serial E3 line to Essen).

6WiN connects to other IPv6 networks as follows:

- to 6NET(native), GEANT(tunnel) and DE-CIX(native) in Frankfurt
- to BCIX(native) and T-Nova(native) at Berlin
- to 6bone(tunnel) at Münster

7.2. Renumbering – Why and How

The 6WiN administration will be handed over from JOIN to G-WiN-NOC (from DFN-Verein) in the near future. For the integration of IPv6 into the GWiN the addresses that are now being used within the 6WiN should be used. For a smooth transition from 6WiN to a dual-stack G-WiN this address space must first be freed, which leads to a router renumbering event within 6WiN. The goal is to renumber the 6WiN backbone to 2001:0638:f::/48.

We will follow the steps proposed by Baker [BAKER] for the renumbering procedure of an IPv6 network without a flag day. Since router renumbering (RR) messages as described in [RFC2894] are not and probably will never be usable the task of exchanging the prefixes will have to be accomplished manually. In summary the following 8 steps have to be performed:

1) **Initial condition**: 6WiN is a stable network with the old prefix 2001:638:0::/48

2) Preparation for the renumbering process

2.1) Domain Name Service

The nameserver *ns.join.uni-muenster.de* as well as secondary nameservers and the delegation for the reverse zone managed by *deneb.dfn.de* are updated.

2.2) Mechanisms for address assignment to interfaces

Since we are renumbering a backbone there is no DHCP or SLAC running. Nor is Prefix Delegation being used.

3) Configuring network elements for the new prefix

4) Adding new host addresses (There are not really any “hosts” within 6WiN)

5) Stable use of either prefix

6) Transition from use of the old prefix to the new prefix

6.1) Transition of DNS service to the new prefix

6.2) Transition to the use of new addresses

7) Removing the old prefix

8) **Final condition**: 6WiN is a stable network with the new prefix 2001:638:f::/48

7.3. Renumbering Steps

These steps require the following work in detail.

7.3.1. Preparing the DNS

To prepare the Domain Name Service (DNS) for quick renumbering most likely the default timers have to be adjusted.

The DNS uses long time caches and seldom updates (to secondary nameservers) to decrease the amount of DNS traffic. During a renumbering event these values are too high (days and weeks) for a short transition phase. Therefore the timers have to be readjusted to minutes and hours.

Usually this is simply done by adjusting the timers in the SOA record of the zone file. In our scenario we have a special case, where the reverse zone for the prefix we use (2001:638:0::/48) is delegated to a subsequent nameserver and the zones f.0.0.0.8.3.6.0.1.0.0.2.ip6.[arpa|int]. have their own zone file. (This is due to the 6WiN still being run by the JOIN team who does not have the authority for the whole zone.)

7.3.1.1. Setting up the new DNS zone

Here the master for the zone 8.3.6.0.1.0.0.2.ip6.[arpa|int]. is the NS deneb.dfn.de and it is delegating the zone 0.0.0.0.8.3.6.0.1.0.0.2.ip6.[arpa|int]. to NS ns.join.uni-muenster.de.

Some special actions have to take place beforehand in this scenario.

- configure a new zone and zone file on *ns.join.uni-muenster.de* for the new prefix
- contact the hostmaster of *deneb.dfn.de* and request the delegation of the new zone
- contact the hostmaster of potential secondary servers to set up secondary service for the new zone

In contrast to action one - which is done by the local administration - actions two and three involve third parties. The response times of these parties are unpredictable which might result in some serious delay of the renumbering process. To make sure the planned time schedule is feasible these steps should be taken very early.

Adding the new zones:

named.conf:

```
zone "0.0.0.0.8.3.6.0.1.0.0.2.IP6.INT"{
                type master;
                file "db.6win.ip6.arpa";
                allow-transfer {::ffff:195.30.0.29/128; 2001:608::/64; };
        };
        zone "0.0.0.0.8.3.6.0.1.0.0.2.IP6.ARPA"{
                type master;
                file "db.6win.ip6.arpa";
                allow-transfer {::ffff:195.30.0.29/128; 2001:608::/64; };
        };
        zone "f.0.0.0.8.3.6.0.1.0.0.2.IP6.INT"{                          #new
                type master;                                             #new
                file "db.6win-new.ip6.arpa";                             #new
                allow-transfer {::ffff:195.30.0.29/128; 2001:608::/64; }; #new
        };                                                               #new
        zone "f.0.0.0.8.3.6.0.1.0.0.2.IP6.ARPA"{                         #new
                type master;                                             #new
                file "db.6win-new.ip6.arpa";                             #new
                allow-transfer {::ffff:195.30.0.29/128; 2001:608::/64; }; #new
        };                                  #new
```

The new zone file is just a renamed copy of the old one. This is an advantage in this scenario. Instead of editing the existing zone file and duplicating every entry to add all the new prefix information we only have to set up a new zone file with a new prefix, resp. a new ORIGIN.

This situation can actually serve as the source for the advice to use separate zone files for every prefix used in the backbone. Instead of delegating this zone to another nameserver one could also use include-files (which can be loaded with a given ORIGIN, too).

7.3.1.2. Adjust DNS timers

DNS uses several timers to distribute its Resource Records to requesting clients. These could either be secondary or caching nameservers. Generally the refresh timers (see RFC 1035) for updating the secondary nameservers are set to very small values but might still last several days.

In order to ensure a quick update of the nameserver the refresh value should be set to a few hours. Usually one now has to wait at least "old refresh timer value" seconds before beginning the renumbering process, because in the worst case the secondaries will wait for that long before they will be reconfigured with the new value. However, newer nameservers (in this case BIND version 9) offer a NOTIFY feature, which will enable the primary nameserver to notify the secondary

nameservers whenever a zone has changed and will force them to update that zone. This way, the shorter timers are spread instantaneously to the secondaries.

Much more important is the “Time To Live” timer (TTL) which defines how long a caching (resp. recursive) nameserver should cache a Resource Record (RR) after querying it from a primary or secondary nameserver. There is no way to updat this values in the caching nameservice, so one definitely has to wait until these timers run out. Commonly the TTL is set to one or more days. So usually one has to readjust this timer to some hours or minutes and has to wait until the old TTL in the caching nameserver has run out before removing the old prefix base addresses during the renumbering process. As this step happens quite late in the whole renumbering process one might even live with the fact, that it may take some days before the caches are cleared.

In our scenario there was no need to "adjust" the DNS timers, as we set up a new zone for the renumbering progress and could set appropriate, small values right from the start. Again, this could be seen as an advantage of setting up a separate zone file for backbone prefixes. It might even outweigh the disadvantage that we had to wait for some hostmasters to react; waiting for the timeout of the TTLs could take longer.

As a matter of fact when we send out our request for the changes regarding the new zone, the DFN hostmaster (of the delegating NS) set up the new delegation within five hours.

The hostmaster of the secondary nameserver even reacted after 29 minutes. This is in fact shorter than the TTL we had set for the old prefixes zone (1 day).

Other timers used by DNS are "retry", "expire" and "negative caching" timers. Retry and negative caching timers are usually set to very short values (within minutes) and shouldn't be our concern in this case. The expire time is often set to a very long time span (four weeks) but is also out of scope for us, as it is only important for secondary nameservers in case of failure of the primary. Nevertheless some of the timers depend on each other. E.g. retry should be smaller than refresh.

One might consider to adjust these values accordingly when changing refresh and TTL timers.

In our scenario we used the following values:

Head of zone file for 0.0.0.0.8.3.6.0.1.0.0.2.ip6.arpa.:

```
$TTL 86400              ; time to live (1 day)
@               IN      SOA     ns.join.uni-muenster.de.        join.uni-muenster.de. (
                        2004111804 ; serial
                        21600 ; refresh (8 hours)
                        3600 ; retry (1 hour)
                        604800 ; expire (7 days)
                        1800 ; minimum (30 minutes)
                        )
```

Head of zone file for f.0.0.0.8.3.6.0.1.0.0.2.ip6.arpa.:

```
$TTL 7200               ; time to live (2 hours)
@               IN      SOA     ns.join.uni-muenster.de.        join.uni-muenster.de. (
                        2004111805 ; serial
```

```
5400 ; refresh (90 minutes)
1800 ; retry (30 minutes)
14400 ; expire (6 hours)
1800 ; minimum (30 minutes)
)
```

7.3.2. Add the new prefix

7.3.2.1. Theoretical plan

We add the new prefix 2001:638:f::/48 to the interfaces of the 6WiN backbone routers.

First we add the new prefix to Hamburg, second to Essen and Erlangen. If no unexpected problems arise we continue with the critical (because connected to other networks) routers Berlin and Frankfurt. For each router we have to change the following settings (see Appendix A for a more complete router configuration):

a) interface LoopbackX

Add new address to the loopback interface.

i.e.: ipv6 address 2001:638:f:600::1/128

b) interface FastEthernetX/Y

Update possible native IPv6 connections to universities on-site.

i.e.: ipv6 address 2001:638:f:600::601:1/112

c) interface SerialX/Y

Add new addresses to point-to-point connections to other 6WiN backbone routers.

i.e.: ipv6 address 2001:638:f:600::b:1/128

d) interface TunnelXYZ

Add new addresses to IPv6-in-IPv4 tunnels to customers.

i.e.: ipv6 address 2001:638:f:600::701:1/112

e) Section “router isis”:

Add new prefix for IS-IS routing.

i.e.: summary-prefix 2001:638:f:600::/56

f) Section “router bgp”

Update neighbors for internal BGP routing.

i.e.: neighbor 2001:638:f:300::1 peer-group internalv6

neighbor 2001:638:f:300::1 description HH-TO-FFM

g) Static routes

Change routes to updated IP addresses.

i.e.: ipv6 route 2001:638:601::/48 2001:638:0:601::601:2

For tunnel interfaces it is possible to avoid using the IP with the old prefix by using the following command (this is a Cisco specific feature). In the preperation phase we exchanged some inconsistent configuration entries from interface IP to interface name.

i.e.: ipv6 route 2001:638:601::/48 Tunnel601

This does not work with other interfaces like FastEthernet0/1 because the MAC address resolution of the next hop only works with point-to-point connections.

The steps b), d) and f) require further work on the client side which follows in the second phase.

7.3.2.2. Practical proceeding

1. Test

To execute these changes we copy the running configuration from each router and execute a perl script (see Appendix A) that changes all of the old prefixes to the new ones and outputs a new configuration. The new configuration is copied back to the router. In detail the following commands are used:

```
client> ssh nr-ham1.6win.de
router> copy /verify running-config scp://join@tolot.join.uni-muenster.de/nr-ham1.cis
client> cisco-renumber.pl nr-ham1.cis 2001:638:0 2001:638:f -a
router> copy /verify scp://join@tolot.join.uni-muenster.de/nr-ham1.cis.new running-config
router> exit
```

Result: This approach did not work because simply duplicating all entries including the BGP routing entries led to route flapping. We could not continue and aborted the test.

2. Test a)

This time we execute the changes to the interface IPv6 addresses step by step without interfering with the routing configuration at first.

First we add the new addresses to the *Loopback6* interfaces. In our case these interfaces are passive (*passive-interface Loopback6*) and the new IP addresses are automatically distributed by IS-IS in the backbone.

Second we add the new addresses to the tunnel and native interfaces. These tasks are done by the allready mentioned perl script.

```
router> copy /verify running-config scp://join@tolot.join.uni-muenster.de/nr-ham1.6win.dfn.de.cis
client> cisco-renumber.pl nr-ham1.6win.dfn.de.cis 2001:638:0 2001:638:f -a -o
```

```
router> copy /verify scp://join@tolot.join.uni-muenster.de/~/scripts/renumber/nr-ham1.6win.dfn.de.cis.new running-config
```

The changes to the tunnel and native interfaces get visible only after modifying the IS-IS configuration (*summary-prefix*).

```
nr-ham1(conf)> router isis
nr-ham1(conf)> address-family ipv6
nr-ham1(conf)> summary-prefix 2001:638:f:600::/56
nr-ham1(conf)> exit-address-family
nr-ham1(conf)> ipv6 route 2001:638:f:600::/56 Null0 254
```

At last we change the IBGP routing entries. For each router we add a new neighbour entry to the existing peering group *internalv6.*

```
nr-ham1(conf)> router bgp 680
nr-ham1(conf)> neighbor 2001:638:f:300::1 peer-group internalv6
nr-ham1(conf)> neighbor 2001:638:f:300::1 description HH-to-FFM
nr-ham1(conf)> address-family ipv4
nr-ham1(conf)> no neighbor 2001:638:f:300::1 activate
nr-ham1(conf)> exit-address-family
nr-ham1(conf)> address-family ipv6
nr-ham1(conf)> neighbor 2001:638:f:300::1 peer-group internalv6
nr-ham1(conf)> exit-address-family
nr-ham1(conf)> address-family ipv6 multicast
nr-ham1(conf)> neighbor 2001:638:0:300::1 peer-group internalv6
nr-ham1(conf)> exit-address-family
```

Result: The routers could not initiate a BGP peering connection with the new IPv6 addresses because the source address of the BGP packets is configured as the *Loopback6* interface. This interface has two IPv6 addresses where of the wrong one (the old one) was selected.

```
Nov 30 15:35:29.959 MET: BGP: 2001:638:F:500::1 open active, local address 2001:638:0:600::1
Nov 30 15:35:29.971 MET: BGP: 2001:638:F:500::1 open failed: Connection refused by remote hostno
debug
```

```
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2001:638:0:300::1
                4   680   39893   18707    41169    0    0 1d01h         459
2001:638:0:500::1
                4   680   24933   18711    41169    0    0 1d01h         104
2001:638:0:800::1
                4   680   37925   18708    41169    0    0 1d01h         188
2001:638:0:A00::1
                4   680   18706   18707    41169    0    0 1d01h           2
```

```
    2001:638:F:500::1
                    4   680       0       0       0    0    0 never    Active
```

To get BGP to work with the new addresses we have to do some more work.

2. Test b)

This time we create an additional loopback interface *Loopback66* with the new IPv6 address as source for a new peering group *internalv66* which uses the router's IPv6 addresses out of the new prefix.

First we create the new loopback interface.

```
nr-ham1(conf)> interface Loopback66
nr-ham1(conf)> no ip address
nr-ham1(conf)> ipv6 address 2001:638:F:300::1/128
nr-ham1(conf)> ipv6 mtu 1480
```

Depending on the router configuration it may be neccessary to make this interface passive.

```
nr-ham1(conf)> router isis
nr-ham1(conf)> passive-interface Loopback66
```

Then we create a new peering group.

```
nr-ham1(conf)> router bgp 680
nr-ham1(conf)> neighbor internalv66 peer-group
nr-ham1(conf)> neighbor internalv66 remote-as 680
nr-ham1(conf)> neighbor internalv66 password 7 081645605F573341
nr-ham1(conf)> neighbor internalv66 update-source Loopback66
nr-ham1(conf)> neighbor 2001:638:F:300::1 peer-group internalv66
nr-ham1(conf)> neighbor 2001:638:F:300::1 description HH-to-FFM-with-F
nr-ham1(conf)> exit
```

Finally we add the new peering entries to BGP.

```
nr-ham1(conf)> router bgp 680
nr-ham1(conf)> address-family ipv4
nr-ham1(conf)> no neighbor internalv66 activate
nr-ham1(conf)> no neighbor 2001:638:F:300::1 activate

nr-ham1(conf)> address-family ipv6
nr-ham1(conf)> neighbor internalv66 activate
nr-ham1(conf)> neighbor internalv66 send-community both
```

```
nr-ham1(conf)> neighbor 2001:638:F:300::1 peer-group internalv66
nr-ham1(conf)> exit-address-family

nr-ham1(conf)> address-family ipv6 multicast
nr-ham1(conf)> neighbor internalv66 activate
nr-ham1(conf)> neighbor internalv66 next-hop-self
nr-ham1(conf)> neighbor internalv66 send-community
nr-ham1(conf)> neighbor 2001:638:F:300::1 peer-group internalv66
nr-ham1(conf)> exit-address-family
```

Results: The peering initiates successfully and works.

```
nr-ham1#show bgp sum
BGP router identifier 193.174.75.237, local AS number 680
BGP table version is 46299, main routing table version 46299
638 network entries using 95062 bytes of memory
1504 path entries using 108288 bytes of memory
682/536 BGP path/bestpath attribute entries using 79112 bytes of memory
1 BGP rrinfo entries using 24 bytes of memory
651 BGP AS-PATH entries using 16488 bytes of memory
81 BGP community entries using 2880 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 301854 total bytes of memory
BGP activity 3928/3225 prefixes, 20060/18426 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2001:638:0:300::1
                4   680   45776   21636    46299    0    0 3d01h         464
2001:638:0:500::1
                4   680   28639   21639    46299    0    0 3d01h         106
2001:638:0:800::1
                4   680   43231   21637    46299    0    0 3d01h         179
2001:638:0:A00::1
                4   680   21635   21636    46299    0    0 3d01h           2
2001:638:F:300::1
                4   680     433      17    46299    0    0 00:12:10      464
2001:638:F:500::1
                4   680    2630    1488    46299    0    0 02:06:43      106
2001:638:F:800::1
                4   680     385      50    46299    0    0 00:36:32      179
2001:638:F:A00::1
                4   680     131     131    46299    0    0 02:01:20        2
% NOTE: This command is deprecated. Please use 'show bgp ipv6 unicast'
```

We do the same with all five routers to get back to a fully meshed network using the new addresses.

7.3.3. Adding the new prefix

The step of adding the prefix to hosts has been omitted because in our backbone there are not any hosts. In our case we have customers which are mainly connected via tunnels and a few which are connected natively to the 6WiN backbone to get IPv6 functionality. We wait till the network stabilizes and inform all 6WiN customers (about 45) to update their network configuration to the new prefix. Because of the great effort and unpredictable time consumption that is required by contacting the customers and partners we first planned to skip this step. To not leave things unfinished we decided in the end to do it anyway. In detail the clients have to do the following things.

The customers are responsible for updating their tunnel end points resp. interfaces.

For case (b) above the client side must change its interface prefix/IP (i.e. 2001:638:0:500::501:2/112 to 2001:638:f:500::501:2/112)

Updating the native connections requires two steps if you do not want a loss of connectivity. First the client side has to add the new IP, then the static route configuration has to be updated for server and client, finally client and server can remove the old IPv6 addresses. Updating the static routes is a vital part of the process which absolutely requires client side action. This is different in comparison to the tunneled connections where client side action in changing the IPv6 address is only neccessary for our network monitoring program. Before starting the renumbering of clients we duplicate the configuration of the monitoring application for the new prefix. In this way it is possible to see immediately when a client reconfigures the interface.

For (d) clients have to change their tunnel end points to the new prefix. (usually accomplished with b))

We have to inform each customer about the new IP address that is to be used and wait for their reaction. This can take a long time. In our case the process took several weeks because contact persons did not answer or needed specific help because they configured the tunnel false.

In addition to that we have to inform external peering partners to change their BGP peering configuration.

For (f) external BGP routing configuration must be changed. In the cases where an interface that sends BGP messages gets renumbered we have to inform the peering partner to update their peering configuration.

We have to negotiate a specified time for the peering configuration update. Affected peering partners are:

- Erlangen: LRZ München
- Frankfurt: IABG Teleport
- Berlin: FHG Fokus -> T-Systems Berkom (because multihop peering over fokus interface)

Need to change the peering to Fokus first and then to Berkom because both peerings use the same hardware interface.

Problems:

- BGP peerings partly need new update-sources. These must be specified as interface names and not as IPv6 addresses. So it is necessary to create new Loopback interfaces.
- If interfaces of multihop BGP peerings get involved, one has to stick to the right order in changing the configuration. You have to change the peerings to the partners with the nearest hop first.

7.3.4. Running with two prefixes

We wait till the new prefix stabilizes throughout the backbone. The progress can be tracked with our system monitoring application.

```
nr-ess1#show bgp sum
BGP router identifier 193.174.75.245, local AS number 680
BGP table version is 40460, main routing table version 40460
637 network entries using 94913 bytes of memory
747 path entries using 53784 bytes of memory
676/536 BGP path/bestpath attribute entries using 78416 bytes of memory
643 BGP AS-PATH entries using 16248 bytes of memory
84 BGP community entries using 2912 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 246273 total bytes of memory
BGP activity 3558/2851 prefixes, 21710/20893 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2001:638:0:300::1
                4   680   37619   23623        0    0    0 00:27:10 Active
2001:638:0:600::1
                4   680   17611   23331        0    0    0 02:18:20 Active
2001:638:0:800::1
                4   680   36221   23849        0    0    0 00:28:42 Active
2001:638:0:A00::1
                4   680   17721   23466        0    0    0 00:28:07 Active
2001:638:F:300::1
                4   680    8535    5499    40460    0    0 2d23h         464
2001:638:F:600::1
                4   680    7162   10022    40460    0    0 4d00h           2
2001:638:F:800::1
                4   680    7463    5498    40460    0    0 2d23h         172
2001:638:F:A00::1
                4   680    4552    5889    40460    0    0 3d03h           2
2001:638:500:1::1
                4   680   26113   49846    40460    0    0 00:10:39      104
% NOTE: This command is deprecated. Please use 'show bgp ipv6 unicast'
```

7.3.5. Update DNS entries

The DNS entries of the primary name server get updated.

```
$TTL 7200;  (old 86400)
@               IN      SOA     ns.join.uni-muenster.de.        join.uni-muenster.de. (
                        2004120301 ; serial
                        5400 ; refresh  (old 21600)
                        1800 ; retry (old 3600)
                        14400 ; expire (old 604800)
                        1800 ; minimum
                        )
;
; TXT Resource Records
;
;
                IN      NS      ns.join.uni-muenster.de.
                IN      NS      ws-ber1.win-ip.dfn.de.
;
;
; $ORIGIN 6win.de
;
tolot           IN  A           128.176.191.6
                IN  AAAA        2001:638:500:101:2e0:81ff:fe24:37c6
www             IN  CNAME       tolot
E-to-MS.n34     IN  AAAA        2001:638:0:500::3:1
MS-to-E.n34     IN  AAAA        2001:638:0:500::3:2
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
nr-fra1         IN  AAAA        2001:638:f:300::1
fra1            IN  CNAME       nr-fra1
nr-ess1         IN  AAAA        2001:638:f:500::1
ess1            IN  CNAME       nr-ess1
nr-ham1         IN  AAAA        2001:638:f:600::1
ham1            IN  CNAME       nr-ham1
nr-ber1         IN  AAAA        2001:638:f:800::1
ber1            IN  CNAME       nr-ber1
nr-erl1         IN  AAAA        2001:638:f:a00::1
erl1            IN  CNAME       nr-erl1
```

After allowing a specific time (in our case seconds) the host names are automatically resolved to the new IP address throughout the network.

Problem:

- In the worst case you have to schedule that the name resolves to the new IP address after the set expire time.

7.3.6. Removing the old prefix

Once everything works well with the new prefix the old one is removed from the network.

First we remove the old BGP peerings.

```
nr-ham1(conf)> router bgp 680
nr-ham1(conf)> no neighbor internalv6 peer-group
nr-ham1(conf)> no neighbor internalv6 remote-as 680
nr-ham1(conf)> no neighbor internalv6 password 7 081645605F573341
nr-ham1(conf)> no neighbor internalv6 update-source Loopback6
nr-ham1(conf)> no neighbor 2001:638:0:300::1 peer-group internalv6
nr-ham1(conf)> no neighbor 2001:638:0:300::1 description HH-to-FFM
nr-ham1(conf)> no neighbor 2001:638:0:500::1 peer-group internalv6
nr-ham1(conf)> no neighbor 2001:638:0:500::1 description HH-to-E
nr-ham1(conf)> no neighbor 2001:638:0:800::1 peer-group internalv6
nr-ham1(conf)> no neighbor 2001:638:0:800::1 description HH-to-B
nr-ham1(conf)> no neighbor 2001:638:0:A00::1 peer-group internalv6
nr-ham1(conf)> no neighbor 2001:638:0:A00::1 description HH-to-ERL
```

Second we remove the routers Loopback6 interface and the SerialX/Y connections with the old IP addresses.

```
nr-ham1(config)> no interface Loopback6
nr-ham1(config)> interface Serial1/0
nr-ham1(config-if)> no ipv6 address 2001:638:0:600::B:1/128
nr-ham1(config-if)> exit
nr-ham1(config)> interface Serial1/1
nr-ham1(config-if)> no ipv6 address 2001:638:0:600::E:1/128
nr-ham1(config-if)> exit
nr-ham1(config)> exit
```

Result: When removing the old prefix from the Serial interfaces routes to clients in the old prefix do not work anymore.

7.4. Comments on problems observed and specific areas for attention

Particular areas to pay attention to include:

- If you are using some kind of network management or monitoring software you have to update its configuration according to the new interface IP addresses in every step.

- Additionally prefix announcements to global Internet should be started after security measures as firewalls have been setup for the new prefix.
- To simplify the renumbering process one should exchange hard coded IP addresses to DNS names as often as possible. If the hardware supports it the amount of manual work is reduced mainly to the reconfiguration of the DNS server.
- Multicast has problems with static routes using interface names (RPF does not work). So renumbering in a multicast enabled backbone is more difficult because you can't exchange the IPv6 addresses and interface names.
- Finally you can say it's not easy to renumber an IPv6 backbone. There is no difference to renumbering an IPv4 backbone.

8. Impact of Renumbering on Network Management Platforms (PSNC)

PSNC is examining useful management tools for renumbering and the impact of renumbering process on management platforms.

The tools might also be used as per [BAKER]: “utilities such as netstat and network analyzers can be used to determine if any existing connections to the host are still using the address from the old prefix for that host.”

8.1. Results

PSNC is evaluating two management platforms:

- HP OpenView Network Node Manager 7.50 - tests for Solaris and Linux versions
- CiscoWorks Campus Manager 4.0 (beta) - tests for this platform are under preparation

8.1.1. Tests for HP OpenView Network Node Manager 7.50

8.1.1.1. Description

HP OpenView Network Node Manager Advanced Edition discovers the existence of IPv6 devices, creates a map showing layer 3 IPv6 device connectivity, then monitors the status of each device. To monitor the address status of a device, NNM uses an IPv6 ping rather than using SNMP requests. NNM considers a device to be down if it doesn't respond to an IPv6 ping.

IPv6 functionality is implemented in HP OpenView Network Node Manager as “the Advanced Routing Smart Plug-in”.

8.1.1.2. Test environment

We perform tests for two versions of HP OpenView: for Linux (as the probably most popular platform) and Solaris.

Management workstations:

- Solaris version was installed on a SunBlade 2000 workstation with Solaris 9

- Tests of the Linux version are not ready, because it seems that HP only supports RedHat Linux (which is available only as commercial software); attempting to install it on a similar Linux distribution (Fedora) failed.

Network:

The topology of the network testbed is presented below.

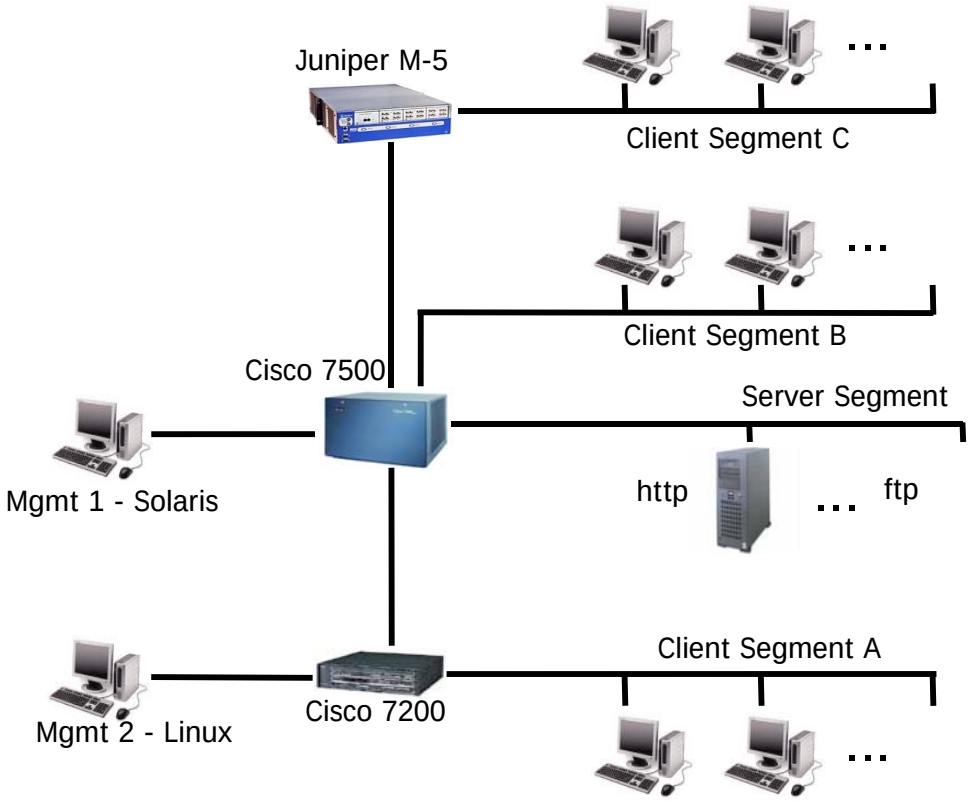

Figure 8-1: PSNC management testbed

In each Client Segment: A, B and C there are different IPv6 prefixes advertised from the router; moreover some of the hosts have additional manually configured addresses.

In the Server segment servers have manually configured IPv6 addresses.

8.1.1.3. Test descriptions

Network discovery:

When you run IPv6 discovery, Extended Topology discovers global, site-local, and link-local addresses. The management station and all routers must be dual-stacked for IPv6 discovery to function properly. Also IPv6 MIBs must be loaded.

Before running IPv6 discovery, IPv6 must be enabled via running the script:

UNIX:`$OV_BIN/setupExtTopo.ovpl`

All IPv6 end node prefixes must be entered manually in appropriate configuration seed file:

- `IPv6Seed.conf` – IPv4 address (recommended option for routers) and IPv6 address and Prefix-Length (recommended option for non-routers)

- `IPv6Prefix.conf` – this file is used to give a user-friendly name to your prefix groups once they are discovered.

The whole network was discovered, including routers and servers segment. The visualization of network is available via accessing “Dynamic Views” using Java interface and the web browser:

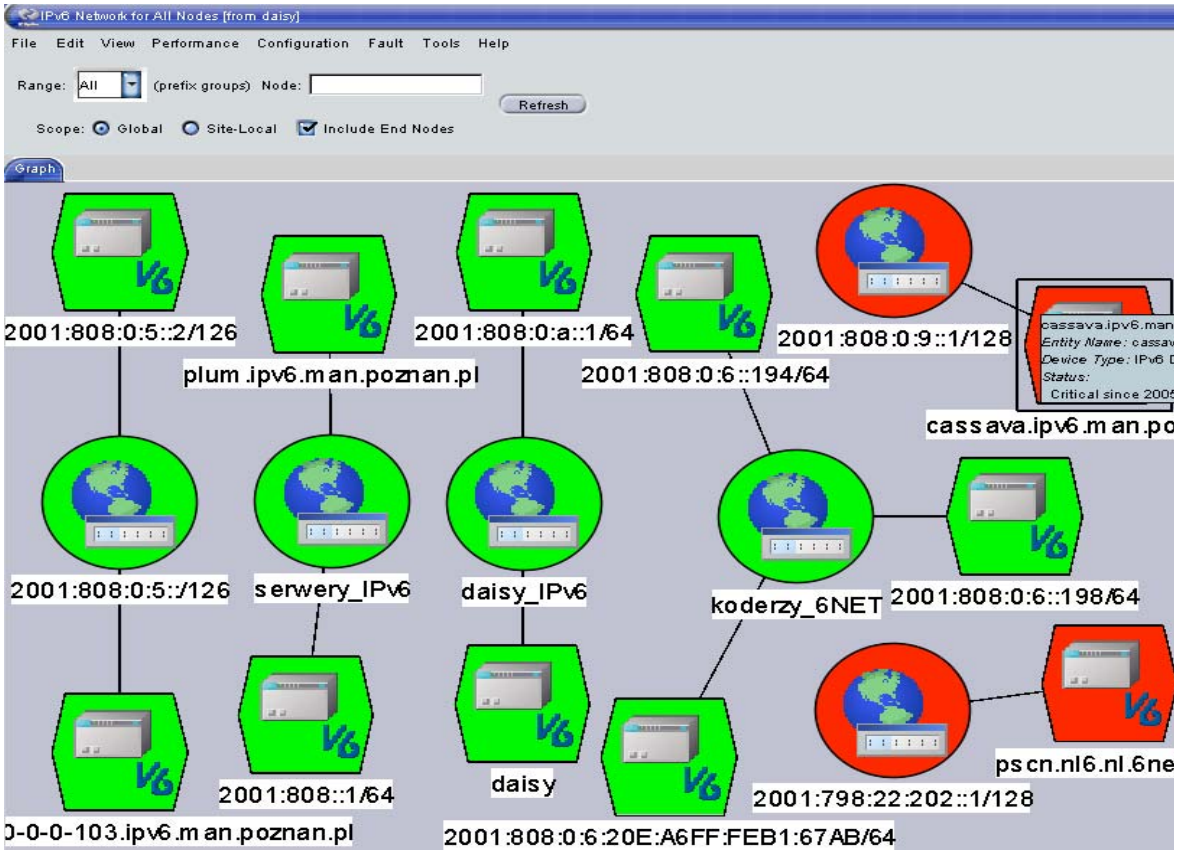

Figure 8-2: Visualization of part of PSNC network testbed in dynamic views

8.1.1.4. Test results and remarks

The following observations were made:

- All hosts within different servers were correctly discovered

- Some of the hosts from the same LAN segment are not visible together in the same LAN segment on the map

- Routers were not correctly discovered; they were visible as hosts and not as routers; however they have SNMP support in MIBs for IPv6; this needs more investigation
- Hosts which are not entered in `IPv6Seed.conf` should also be discovered, but they were not; the reason is that the management station can only discover them from the routing table from routers, but the routers are not correctly discovered
- Information about changes in configuration cannot be performed automatically, it can be done only via rediscovering
- Discovering is a very long process (it also discovers the whole information for layer-2 devices); for a few segments in the network testbed the process took about 20 minutes or more
- Hosts without an SNMP agent with two IP addresses on an interface are discovered twice, as two hosts

8.1.1.5. Final conclusions

We can draw the following conclusions on the network management platform impact on the renumbering process:

- HP OpenView Network Node Manager can easily discover nodes and confirm if the renumbering process was correct
- Discovering in renumbered networks is a long process and it has to be triggered manually
- Some changes in reconfiguration of management workstation must be performed manually (or via prepared scripts)
- Hosts and devices without SNMP support for IPv6 in MIBs are visible as one instance of each configured address on an interface
- Routers and devices with SNMP support for IPv6 in MIBs are queried using IPv4; this communication is not disturbed during the IPv6 renumbering process

9. Renumbering trials with Management/supervision tools (Loria/Inria)

It is inconceivable in wide-area networks for a network administrator to physically supervise each host; this is why one sets up an architecture of supervision and management based on many tools. This is called the management plane.

Once set up, a network management platform allows the network administrator to monitor each node on their network and to be informed of many problems without having to leave their workstation.

One of the main advantages of IPv6 is Address Autoconfiguration, which in theory helps to ease the renumbering process on an IPv6 network. But what happens to the management plane during and after this?

Specifically:

- What happens to the management plane after and during a renumbering event?
- Do the applications deployed still work or not? Which applications work, and which do not?
- What actions are required to correct the problems?

To begin with, we need to identify the problems and service outages and keep in mind the first issues highlighted in [THINK].

Once these are clearly identified, we aim at providing guidelines on how to renumber a network if management/supervision applications are deployed, in which we simply explain what are the required actions, before, during and after the renumbering in order to avoid having applications crashing or service outage.

Finally, for some applications, after the renumbering, there could be problems linked to data continuity (the past statistic addresses and/or data collected may be invalid or in some way corrupted or simply lost). We propose to write scripts and develop (if needed) service extensions for the applications deployed in our testbed in order to ensure data continuity.

9.1. Testbed

Our testbed is quite simple, as shown below. The network renumbered doesn't have any dual stack, it is an IPv6-only network, but, we may from time to time use IPv4 on this network for some tests.

All hosts are running a GNU/Linux 2.6.10 kernel with USAGI IPv6 stack.

Asterix, our router, is a PC running a Debian GNU/Linux distribution, and uses both radvd and Netfilter (ip6tables).

This host will be the main enabler in the renumbering of the network. The procedure we follow is described in [BAKER]. If we summarize it, we begin by adapting the TTL and lifetime of the old prefix and include the new one in the routing architecture. Then we announce the new prefix so that we obtain a stable situation with both prefixes available and usable. The next step is advertise the old prefix as obsolete, and then only announce the new one and remove all reference to the old one (including the routing configuration and update of the reverse DNS). The final outcome is then a stable situation with only the new prefix in operation.

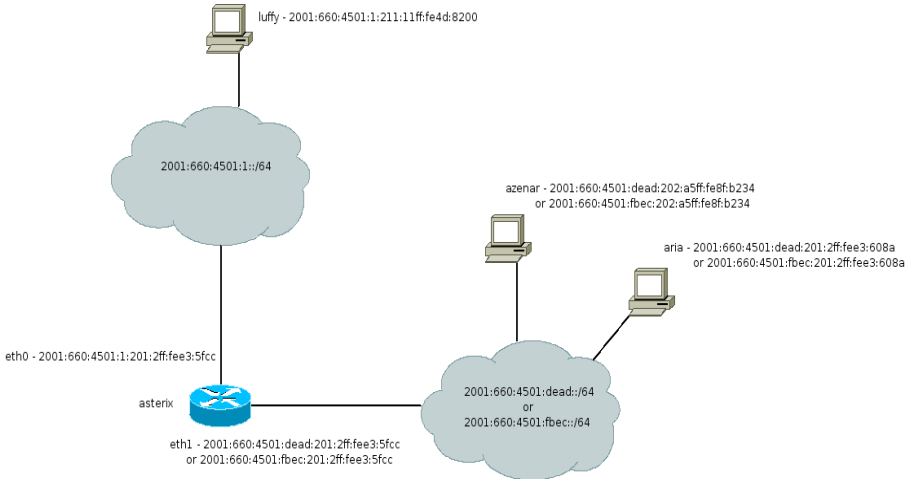

Figure 9-1: Loria network testbed

We noticed during these tests that one of the most important variables for such scenarios is the prefix TTLs and lifetimes. Actually, the hosts renumbered keep the global address corresponding to the announced prefixes as long as the timer associated hasn't expired (which means advertised lifetime seconds after the last Router Advertisement) and will do that even if the prefix has been announced as obsolete before stopping the advertisement.

This means that, if D is the day we want to renumber a network and the default lifetime of prefixes is 1 day, we will keep the corresponding addresses on host interfaces until D+1. If we want this transition period to be shorter, we have to modify this parameter at least at D-1, so that the modification could be taken in account by the hosts.

For convenience, we set all announced lifetimes on our testbed to 6 minutes.

Both hosts (luffy and azenar) are at the same time monitoring and monitored stations which monitor each other, but whereas azenar is the only one which is renumbered, luffy's address isn't changing. This way, we can see what happens in most of the cases. Finally, in the renumbered network, azenar and aria monitor each other, and of course are both renumbered.

We deployed the same management/supervision applications on luffy, aria and azenar, which are NetSNMP, Ntop, MRTG and Nagios.

The renumbered hosts are configured to use Stateless Address Autoconfiguration (SLAAC) for practical purposes, instead of DHCPv6.

9.2. Tests and Results

In this section, we investigate the management/supervision plane after a renumbering event, and try to answer the questions posed above.

9.2.1. Renumbered station monitors non-renumbered hosts

On our testbed, this means azenar or aria monitors luffy and is renumbered.

In this case, there isn't any problem concerning the application configurations. As the monitored hosts' addresses do not change, as long as the routing is correctly configured, the management plane is still usable.

The only problem which could occur is linked to security and eventual ACL or restrictions concerning the access on the hosts monitored. For example, as the prefix of the monitoring station has changed, if we set up NetSNMP's daemon by accepting only connections coming from the old prefix, we need to update this and restart the service.

9.2.2. Non-renumbered station monitors renumbered hosts

On our testbed, this means luffy monitors azenar or/and aria which is renumbered. In the following sections, we will detail each tool and the first conclusions we made.

9.2.2.1. Ntop

If Ntop's web interface is queried on the new address, it still works fine.

But there is one problem: the data and statistics collected aren't updated anymore, and we need to restart the service to make it work again, which means losing the data collected before and during

the renumbering. This problem is caused by the fact that when starting, Ntop determines the local address by using the NIC's settings.

9.2.2.2. NetSNMP

As long as SET and GET operations are done on the new address, there is no problem with NetSNMP, whatever the version of the protocol used is (v1, v2 or v3/USM).

Nevertheless, we have to note that only the basic IPv6 MIB (IPV6-MIB) is supported under GNU/Linux at the moment, the MIBs which take care of the TCP or UDP tables (IPV6-TCP-MIB or IPV6-UDP-MIB) aren't working in the current release.

9.2.2.3. MRTG

In the same way as Ntop, if the web interface is queried on the new address and for the correct HTML indexes, there is no problem for viewing the statistics.

There isn't any problem either if the renumbered hosts use the localhost address ([::1]) to monitor themselves.

For the hosts which have changed their address during the renumbering phase, we need to reconfigure MRTG to deal with the modifications. By doing this, we need to manually make modifications on the configuration files (or use the cfgmaker tool, but this still implies making changes by hand).

But, these operations have also an impact on data and on the generated graphics. As the filenames of these are depending on the address of the monitored host, they change and then the data are reseted.

To ensure data continuity, the graphics' filenames have to be renamed according to the new addresses, and the HTML index has also to be modified in order to integrate these modifications.

A script has been written to automatically do all the operations:

```
#!/bin/bash
#
# MRTG Continuity after renumbering script
#
#Usage : ./mrtg_continuity.sh <old_prefix> <new_prefix> <PATH_to_MRTG_config_files>
#


# The script takes 3 parameters
if [ $# -ne 3 ]
then
  echo
  echo "Usage : $0 <old_prefix> <new_prefix> <PATH_to_MRTG_config_files>"
  exit 0
fi


# Step 1 : rename files
```

```
echo "Step 1 : renaming files in /var/www/mrtg"
for file in `ls /var/www/mrtg/*$1*`
do
        new=`echo $file | sed s/$1/$2/g`
        mv ${file} $new
done

# Step 2 : edit HTML file
echo "Step 2 : replacing old filenames in HTML files"
for file in `ls /var/www/mrtg/*$2*.html`
do
        sed s/$1/$2/g $file > $file\_modified
        mv $file\_modified $file
done

# Step 3 : edit configuration file
echo "Step 3 : replacing old prefix by the new one in configuration files"
for file in `ls $3/*.cfg`
do
        sed s/$1/$2/g $file > $file\_modified
        mv $file\_modified $file
done

exit 0
```

9.2.2.4. Nagios

As for MRTG, as long as the web interface is queried on the new address, there is no problem, as for a renumbered host monitoring itself by using the localhost address.

For the hosts renumbered which address has then changed, we need to manually edit the configuration file *hosts.cfg* which contains the hosts declarations in Nagios to update the addresses. In the same way, for some plugins, the file *services.cfg* may need an update if for some plugins the hosts addresses appear "in hard" in this file. In order to deal with the modifications, Nagios has to be restarted.

Concerning data continuity, as Nagios uses the hostname defined in the *hosts.cfg* configuration file to identify each host, and as this hostname doesn't change (unless the administrator thinks he should change it, there is no reason linked to the renumbering to modify it), data continuity is assured.

As we did for MRTG, we wrote a little script which performs all the needed actions :

```
#!/bin/bash
#
```

```
# Nagios Continuity after renumbering script
#
#Usage : ./nagios_continuity.sh <old_prefix> <new_prefix> <PATH_to_nagios_config_dir>

# The script takes 3 parameters
if [ $# -ne 3 ]
then
  echo
  echo "Usage : $0 <old_prefix> <new_prefix> <PATH_to_nagios_config_dir>"
  exit 0
fi

echo "Step 1 : Updating hosts adresses in $3/*.cfg"

for file in `ls $3/*.cfg`
do
        sed s/$1/$2/g $file > $file\_modified
        mv $file\_modified $file
done

echo "Step 2 : restarting nagios"

/etc/init.d/nagios restart

exit 0
```

9.2.3. Renumbered station monitors renumbered hosts

9.2.3.1. Using global addresses

This is the last case we have to look at when using global addresses. This situation means in our testbed that aria and azenar monitor each other while the network is renumbered.

For all the tools deployed, the conclusions we can make are the same than these we already made in the previous sections, except for Ntop.

Actually, the difference concerns the statistics update. When the network is renumbered, if the monitoring station uses a dual stack network IPv4 and IPv6, even if the IPv6 address used by Ntop isn't updated according to the new address, the stats are still updated (for both IPv4 and IPv6).

We used the same script to resolve the problems, and then it worked perfectly.

9.2.3.2. Using Link Local Addresses

A solution to avoid having trouble with the renumbering of an IPv6 network would be to use, when possible, Link Local Addresses (LLA). To check if this solution could be usable, we tried to use these LLA on aria and azenar, the two hosts in the renumbered network on our testbed.

What we did is configure MRTG and Nagios so that azenar and aria monitor each other using their LLA, and also tried SET and GET operations on the NetSNMP daemon running on both hosts. Ntop isn't concerned, because there isn't any such configuration for this tool.

Using LLA with NetSNMP works fine, SET and GET operations succeed, before, during and after a network renumbering. Thus, if MRTG is configured to use these LLA, the service suffers no outage and is still operational after and during the renumbering.

Concerning Nagios, it is a little more complicate. Actually, all plugins don't work if using the LLA. Some of them (check_tcp or check_http for example) return an error and a message *Argument Invalid* when using the LLA. As expected, the others (check_snmp for example) work fine before, during and after the renumbering, thanks to the nature of the LLA itself.

ULAs could be similarly tested.

9.3. Further Work

These are only the first results. We still have some tests to do, in order to confirm our conclusions in different situations (at the moment, we are making tests on multihomed networks). We also have to take a look at the problem from the software developer's point of view, and try to provide some hints on how these could evolve in order to avoid some of the problems we highlighted earlier.

Once these are done, the first thing we would like to take care of is the data continuity for Ntop. We don't know yet if this will be done by writing a script or modifying Ntop itself.

Then, we are aiming at providing, either a dependence MIB or a daemon running on the managed hosts and which would alert the management station if something went wrong. For example, if an IPSec based VPN or a videoconferencing session is running, the monitored host may want to keep these active, and would ask the router and/or the managed station to keep for a certain time both prefixes active and usable, or simply alert the management host that it hasn't managed to achieve the renumbering and eventually use a built in database to diagnostic the reason for that.

Finally, we are planning to automatically make the updates required on the management applications after a renumbering has occurred, maybe by using the daemon or MIB mentioned in the second point of this section.

9.4. Conclusions

We have a first set of results, and these are quite frightening: almost everything fails in some way.

Nevertheless, the problems are well known, and it will be possible for us to provide guidelines and examples of solutions, which won't work in all cases and for every configuration, but which could be a good starting point to further research and solutions.

10. Recommendations (Southampton)

Here we offer some general recommendations aimed at specific audiences.

10.1. Recommendations for Network Designers

10.1.1. Tokenised Interface Identifiers for Core Services

'Tokenised' well-known addresses offer the ability to have known addresses for core services without the overhead of running DHCPv6.

Enabled through the 'token' socket I/O directive, SIOCSLIFTOKEN, specifying a token on an interface before it is brought up will mean that the token is used for the interface identifier part of statelessly autoconfigured address as opposed to the EUI-64 address derived from the NIC's MAC.

The FE80::/10 link-local address will still be auto-generated using the EUI-64 address, however any Router Advertisement received in compliant nodes will be prepended to the token to form a node address of <prefix>::<token>. For example, specifying a token of 25 might be appropriate for a site's (or subnet's) SMTP relay.

10.1.2. Use of Anycast within a site; or globally

Beyond the use of tokenised interface identifiers, one recommendation that is emerging is the use of Anycast for core services, whether locally as well-known site-wide anycast addresses independent of allocated provider-assigned addressing, or globally using globally reserved anycast addresses such as used in IPv4 for the 6-to-4 transition mechanism.

For example, specifying a globally recognised anycast address for DNS servers as a shipping default (e.g. in operating systems, or as OS "kickstart" images) would offer a 'least effort guaranteed functionality' for nodes, with address resolution being operational without additional configuration or the presence of a stateless DHCPv6 service locally, particularly when combined with stateless address auto-configuration.

10.1.3. Network Architecture Protection

RFC2072 stated: "Network address translation (NAT) is a valuable technique for renumbering, or even for avoiding the need to renumber significant parts of an enterprise." That is, by 'hiding' the subnet topology and making independent of any connectivity provider the addressing model used within a site, NATs enable renumbering of entire networks because the only device that is renumbered when global addressing changes is the outside edge of the NAT devices.

However, NAT is strongly discouraged in IPv6, not least because NAT devices obscure identity - the basis for permission, authorisation, verification and validation - and thus should not be considered as being available as a solution. A significant reason to deploy IPv6 is to simplify network and application operation by NAT removal, for example to provide true end-to-end connectivity, to make simple the gateway between site and Internet, to encourage 'considered' policy as regards secure access rather than the weak and dangerous defence of hiding behind a NAT. A more detailed discussion of the motivations for 'protecting' the network architecture from NATs can be found in [NAP].

10.1.4. Unique Local Addresses for intra-site interactions

Section 5 of [ULA] suggests that the use of Local IPv6 addresses in a site results in making communication using these addresses independent of renumbering a site's provider based global addresses. It also points out that a renumbering episode is not triggered when merging multiple sites that have deployed centrally assigned unique local addresses as per [ULA-C] because the FC00::/8 ULA prefix assures global uniqueness.

With reference to section 2 of [BAKER], the adoption of ULA to assist in network renumbering can be considered a 'seasoning' of Baker's renumbering procedure: where interaction between local nodes and their services cannot suffer the inherent issues observed when migrating to a new aggregatable global unicast prefix, the use of FC00::/7 unique local addresses may offer an appropriately stable and reliable solution. The use of ULAs in single-site networks (e.g. SOHO) appears straightforward and of immediate benefit with regards renumbering episodes triggered by uplink connectivity changes. How their use scales to multi-site (e.g. Enterprise or use in ISP or transit networks) is not so evident.

The use of ULAs may not necessarily be accompanied by PA addresses. If addresses under a PA global routing prefix are not used, some form of IPv6 NAT or application layer gateway deployment will be required for ULA-only nodes internal to the network to communicate with external nodes that are not part of the same ULA topology, that is destination nodes that are not part of the same administrative domain from which the ULA allocation of the local node is made, nor part of a predetermined routing agreement between two organisations utilising different ULAs for nodes within their own sites. ULAs are not intended to be routed globally.

If addresses under a global routing prefix are also deployed, then nodes will need to cater for being multi-addressed, e.g. follow the principles laid out in RFC3484. The administrator should ideally be able to set local policy such that nodes use ULAs for intranet communications and global addresses for extranet communications. The use of ULAs internally would in principle mitigate global address renumbering of nodes.

ULAs appear to lend themselves particularly well for long-lived sessions whose nature is intra-site, for example local file-store mounts over TCP-mounted NFS: With clients using ULA source addresses to mount file-store using the ULA of an NFS server, both client and server can have their global routing prefix renumbered without consequence to ongoing local connections.

10.1.5. Understanding Equipment Ownership

The question of who owns and administers (also, who is authorised to administer) a site's access router is an issue in some renumbering situations, particularly with scheduling the renumbering episode.

In the enterprise scenarios, the liaison between the end users and remote administrators is likely to be relatively easy; this is less likely to be the case for a SoHo scenario.

This is not likely to be a major issue, however, since SoHo renumbering is likely to only be required if the remote administrators deem it necessary, or if the end user is sufficiently technically competent and decides to renumber their own network.

10.2. Recommendations for Network Administrators

10.2.1. Avoid address literals

It is strongly recommended that administrators avoid using hard-coded address literals wherever possible.

There are many places in the network where IP addresses are embedded as opposed to symbolic names, and finding them all to be updated during a renumbering episode is not a trivial task.

Addresses may be hard-coded in software configuration files or services, in software source-code itself (which is particularly cumbersome if no source is available, e.g. a bespoke utility built to order), in firmware (for example, an access-controlling hardware dongle), or even in hardware, e.g. fixed by DIP switches.

A non-exhaustive list of instances of such addresses was presented above in Section 3.

Some hard-coded IP address information will be held in remote locations, e.g. remote firewalls, DNS glue, etc., increasing the complexity of the search for all instances of the old prefix.

Should symbols be used rather than addresses, administrative ownership of DNS - with due consideration for the TTL of resource records - and other naming services ease this particularly problematic issue of data ownership and validity.

Note that relying on labels for endpoint identification during possible renumbering can be problematic should administrators be lackadaisical in planning for the renumbering episode by stepping down DNS Resource Record Time To Live data.

10.2.2. Plan service data

On routers, it is vital to remember to step-down the Router Advertisement Validity and Preferred times, but with care to ensure that step-down for Validity time is advertised ahead of 2 hours before the renumbering episode is to take place [RFC2462]. This enables stacks to deprecate the old prefix in a timely manner and then subsequently remove the old prefix and associated node-maintained route information at the appropriate time (i.e. as the old prefix genuinely becomes unavailable, not afterward).

It is also recommended that, where possible, administrators plan to 'remove' the old prefix from operational use well in advance of any hard limit imposed, e.g., by up-stream providers.

Should a site be using DHCPv6 for stateful address configuration, be sure to manage the lowering of lease times and prepare for Reconfigure messages [DHCPv6] to be sent (individually) to bound clients.

Where the site has node global address data in the DNS, the Time To Live data bound to resource records concerning nodes that are to be renumbered should be decreased and zone data cache and refresh intervals ramped such that external nodes will be assured of resolving (authoritatively) correct data post-renumbering.

10.2.3. Apply policy before renumbering

So as not to leave oneself open to attack and to be confident that the access control policy is applied from the outset, it is recommended that firewalls, packet filters and other AAA technologies are configured ahead of the new prefix data being applied to routers and hosts.

Note that this may well go against the recommendation made that suggests mandating the use of labels over literals. However, if the firewall supports prefix labels, the policy can be applied more readily to both the old and new prefixes.

10.2.4. Avoid Stateless DHCPv6

Until the work on Stateless DHCPv6 lifetime [DHCLIFE] has stabilised and seen implementation, it is recommended that administrators avoid the use of Stateless DHCPv6 Information-Request options in networks where configuration information is likely to change (such as DNS or NTP server addresses).

The adoption of global anycast identifiers for core services that are not location-dependent would negate this recommendation for the most part.

10.2.5. Confirm monitoring tool behaviour

When relying on DNS labels for identifying nodes to administer, care must be taken to ensure that the complete set of nodes administered are caught. For instance, a set of application servers may share the same DNS label and rely on DNS round-robin for rudimentary load balancing. A network monitoring tool that was configured to monitor just that service that was resolved by address lookup might only capture one of that set of nodes.

10.3. Recommendations for Application Developers

10.3.1. Resolve addresses for each connection attempt

Applications should rarely cache DNS information, instead querying the resolver whenever DNS is required. This leaves the resolver to handle the Time-to-Live of the address records.

Application programmers should only cache DNS information if they are going to be opening a lot of connections in a short amount of time, and do not care about the possibilities of renumbering – this is up to the programmer's judgement.

10.3.2. Bind to wildcard service addresses

Application developers are encouraged to not bind to a specific label (e.g. from a configuration file), particularly as that label will likely only be resolved once (at invocation time) by a naming service.

Rather, it is recommended that all Internet-facing services are configured such that their server sockets bind to the unspecified address.

10.3.3. Use SCTP as a TCP alternative

For applications that require long-lived sessions and therefore desire to survive renumbering episodes in which on-going connections would otherwise fail at the point where the old prefix data was no longer valid, it is recommended that the Stream Control Transmission Protocol [RFC2960] in considered in preference to TCP.

A key enabler of SCTP is its ability to dynamically change the endpoints associated with a live stream [SCTPDAR], meaning that in the stages where both old and new prefixes are available for a participant node, the association between sender and recipient can be extended to include the new address data; with the old data being removed at some point before the old address invalidates and becomes unusable.

10.4. Recommendations for Operating System and Stack Developers

10.4.1. Open TTLs to applications

An extension to the Sockets API that exposes liveness metadata or DNS resource records (e.g. validity time of a resolved symbol at a minimum) would enable application and middleware developers to make a reasoned decision regarding the use of result data from the resolver library.

10.4.2. Adhere to DNS TTL

It is strongly recommended that resolver libraries strictly adhere to DNS Resource Record Time-to-Live data, and that they query for authoritative data when any cached copy has timed-out.

Assuming that the network administrators at the site undergoing a renumbering episode are following the other recommendations in this memo, the authoritative data for the resource records should be such that the updated records should be picked up by all TTL-obeying resolvers in a timely manner.

11. Conclusions

This document has presented a collection of theory (protocols, tools and scenarios), experiments (SOHO and backbone oriented) and an initial set of suggested best practices to help ease the process of IPv6 network renumbering.

11.1. Summary

There are a number of conclusions that can be drawn from the work presented in this document. These include:

- Tools and methods to ease renumbering of an IP network have not been advanced as much as they might have been, most likely because IPv4 offers Provider Independent (PI) address space for large enterprises. The PIER WG began studies in 1996 (or so) but closed shortly thereafter after publishing two RFCs.

- There are some features of IPv6 that lend themselves to easing renumbering (stateless autoconfiguration, DHCP-PD, Router Renumbering) but these still need further deployment trials. In addition, many of the common IP problems remain (e.g. hard-coded IP literals, in local and remote systems).
- Scenarios and triggers for network renumbering appear to be relatively well defined and understood.
- A phased process for IPv6 renumbering without a flag day has been proposed [BAKER]. Trials with 'simple' SOHO scenarios (reported in this document) suggest the procedure is sound. Further work is required in larger enterprise scenarios, where DHCP-PD may assist, for example.
- Network backbone renumbering following [BAKER] was generally sound, but found some issues, e.g. in multi-hop BGP usage (and ordering of changes). The IPv6 renumbering features did not really help with the backbone renumbering process (it seemed as complex as IPv4 renumbering).
- Network management tools (still in their relative infancy for IPv6) are generally not well adapted to handle IPv6 renumbering events. Either the tools need restarting, statistics files need to be 'post-processed', or other undesirable properties would manifest themselves, e.g. multi-addressed devices might appear as multiple nodes on the network. There is an impact on both management and monitoring applications.
- Running an IPv6 link with two (non link-local) prefixes active is quite possible, but a number of issues have been identified in this text. Further experimentation is desirable.
- A set of recommendations has been identified for key target audiences (network designers and administrators, application developers, and operating system/stack developers). These need to be disseminated to those audiences for discussion and progression where possible. Further recommendations may also be drawn.

11.2. Future Work

Renumbering experiments geared towards enterprise networks will be reported in D3.6.2, along with considerations for ISP networks.

A number of actions would be most helpful in advancing the understanding of the practical implications and robustness of IPv6 renumbering. These include:

- An extended survey of the pervasiveness of address literals and steps to avoid their use
- Further validation of address selection at source and destination during various stages of Baker's renumbering procedure [BAKER]
- Validation of RA lifetime expiry and confirmation of prefix removal and effects on existing sessions on various platforms
- Validation of IPv6 Prefix Delegation by DHCP, and of IPv6 Router Renumbering
- Better understanding of the commonalities and differences between renumbering and multi-homing (including new IPv6 shim6 WG proposals)

- Anecdotal experience of IETF participants or RIPE members that have undertaken an IPv6 renumbering exercise, e.g. in the transition from 3FFE::/16 6Bone addresses to production GAU addresses
- The use of ULA addresses in parallel with global addresses, to offer a stable addressing environment in the face of a renumbering event (but at what cost…?)
- New tools could be developed to assist the renumbering process, e.g. a Netflow-based tool to detect accesses to 'expiring' prefixes.

We argue in [THINK] that there may be a case to be made to reopen the PIER WG in the new context of IPv6, although that group has (apparently) not been active since 1997. This would require a BoF at a future IETF event. This would be best advanced if and when [THINK] is adopted as an IPv6 v6ops WG item.

The authors would be very happy to receive feedback and suggestions for improvements to the content of this cookbook, with a view to improving its usefulness and applicability, such that further point revisions can be issued beyond the lifetime of the 6NET project if necessary. Please send such comments to the editor, Tim Chown at *tjc@ecs.soton.ac.uk*.

12. References

[6NET] The 6NET Project, http://www.6net.org/

[6NET-D224] "Final Updated IPv4 to IPv6 migration Cookbook for organisational/ISP (NREN) and backbone networks", 6NET Project deliverable D2.2.4, http://www.6net.org/publications/deliverables/D2.2.4.pdf

[6NET-D234] "Pre-final IPv4 to IPv6 transition Cookbook for end site networks/universities", 6NET Project deliverable D2.3.4, http://www.6net.org/publications/deliverables/D2.3.4.pdf (to be updated with v2 by June 2005)

[6NET-D312] "IPv6 cookbook for routing, DNS, intra-domain multicast, inter-domain multicast and security, 2nd Version", 6NET Project deliverable D3.1.2, http://www.6net.org/publications/deliverables/D3.1.2v2.pdf

[6NET-D622] "Operational procedures for secured management with transition mechanisms", 6NET Project deliverable D6.2.2, http://www.6net.org/publications/deliverables/D6.2.2.pdf

[6NET-D624] "Final report on IPv6 management tools, developments and tests", 6NET Project deliverable D6.2.4, http://www.6net.org/publications/deliverables/D6.2.4.pdf

[6NET-D633] "Final report on IPv6 management and monitoring architecture design, tools, and operational procedures - Recommendations", 6NET Project deliverable D6.3.2, http://www.6net.org/publications/deliverables/D6.3.3.pdf

[ANYOP] Abley, J. and K. Lindqvist, "Operation of Anycast Services", draft-ietf-grow-anycast-00 (work in progress), February 2005.

[BAKER] Baker, F., "Procedures for Renumbering an IPv6 Network without a Flag Day", draft-ietf-v6ops-renumbering-procedure-05 (work in progress), March 2005.

[CAMPUS] Chown, T., "IPv6 Campus Transition Scenario Description and Analysis", draft-chown-v6ops-campus-transition-01 (work in progress), October 2004.

[DHCLIFE] Venaas, S. and T. Chown, "Information Refresh Time Option for DHCPv6", draft-ietf-dhc-lifetime-03 (worjk in progress, in RFC editor queue), January 2005.

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

[DNSRES] Jeong, J., "IPv6 Host Configuration of DNS Server Information Approaches", draft-ietf-dnsop-ipv6-dns-configuration-06 (work in progress), May 2005.

[HIPARCH] Moskowitz, R., "Host Identity Protocol Architecture", draft-ietf-hip-arch-02 (work in progress, under IESG evaluation), January 2005.

[HUSTON] Huston, G., "Architectural Approaches to Multi-Homing for IPv6", draft-ietf-multi6-architecture-04 (work in progress, in RFC editor queue), February 2005.

[ISATAP] Templin, F., Gleeson, T., Talwar, M., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", draft-ietf-ngtrans-isatap-24 (work in progress), January 2005.

[MULTI6] IETF Multi6 WG, http://www.ietf.org/html.charters/multi6-charter.html

[NAP] Velde, G., "IPv6 Network Architecture Protection", draft-ietf-v6ops-nap-00 (work in progress), March 2005.

[NEMO] Ernst, T. and H. Lach, "Network Mobility Support Terminology", draft-ietf-nemo-terminology-03 (work in progress), February 2005.

[RFC1546] Partridge, C., Mendez, T., and W. Milliken, "Host Anycasting Service", RFC 1546, November 1993.

[RFC1916] Berkowitz, H., Ferguson, P., Leland, W., and P. Nesser, "Enterprise Renumbering: Experience and Information Solicitation", RFC 1916, February 1996.

[RFC2071] Ferguson, P. and H. Berkowitz, "Network Renumbering Overview: Why would I want it and what is it anyway?", RFC 2071, January 1997.

[RFC2072] Berkowitz, H., "Router Renumbering Guide", RFC 2072, January 1997.

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461 (under revision as draft-ietf-ipv6-2461bis-03), December 1998.

[RFC2462] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462 (under revision as draft-ietf-ipv6-rfc2462bis-08 under IESG evaluation), December 1998.

[RFC2526] Johnson, D. and S. Deering, "Reserved IPv6 Subnet Anycast Addresses", RFC 2526, March 1999.

[RFC2894] Crawford, M., "Router Renumbering for IPv6", RFC 2894, August 2000.

[RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, February 2000.

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

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

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

[RFC3177] IAB and IESG, "IAB/IESG Recommendations on IPv6 Address Allocations to Sites", RFC 3177, September 2001.

[RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, August 2002.

[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.

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

[RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

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

[RFC3701] Fink, R. and R. Hinden, "6bone (IPv6 Testing Address Allocation) Phaseout", RFC 3701, March 2004.

[RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.

[RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", RFC 3956, November 2004.

[SCTPDAR] Stewart, R., "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", draft-ietf-tsvwg-addip-sctp-12 (work in progress), June 2005.

[SRCPOL] Matsumoto, A., "Source Address Selection Policy Distribution for Multihoming", draft-arifumi-multi6-sas-policy-dist-00 (work in progress), October 2004.

[THINK] Chown, T., Ford, A., Thompson, M. and S. Venaas, “Things to think about when Renumbering an IPv6 network”, draft-chown-v6ops-renumber-thinkabout-02 (work in progress), May 2005.

[ULA] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", draft-ietf-ipv6-unique-local-addr-09 (work in progress, in RFC editor queue), January 2005.

[ULA-C] Hinden, R. and B. Haberman, "Centrally Assigned Unique Local IPv6 Unicast Addresses", draft-ietf-ipv6-ula-central-01 (work in progress), February 2005.

[V6OPS] IETF IPv6 Operations WG, http://www.ietf.org/html.charters/v6ops-charter.html

13. APPENDIX A: **Router renumbering script in Perl**

```
#!/usr/bin/perl

# Router renumbering script
#
# Changes all occurences of an old prefix in a Cisco router configuration file to a new prefix.

my ($cfgin, $cfgout);
my ($old_prefix, $newprefix);
my ($mode) = "replace";

if ($#ARGV<2)
{
  print "usage   : $0 <config_file> <old_prefix> <new_prefix> <options>\n";
  print "files   : use '-' to read and write to STDIN or STDOUT\n";
  print "options : -a    Add new prefix to configuration.\n";
  print "            -o     Add only new interface addresses to configuration. Do not change routing commands.\n";
  print "          -r    Replace old prefix in configuration. (Default)\n";
}
else
{
  $cfgin = shift @ARGV;
  $cfgout = "${cfgin}.new";
  if ($cfgin eq "-")
  {
    $cfgin = "&STDIN";
    $cfgout = "&STDOUT";
  }
  $old_prefix = shift @ARGV;
  $new_prefix = shift @ARGV;
  $options = "@ARGV";

  if ($options =~ /-a/)
  {
    warn "Adding new prefix to existing one.\n";
    $mode = "add";
  }
  if ($options =~ /-o/)
  {
    warn "Changing only interface addresses.\n";
```

```
        $mode = $mode.",interfaceonly";
      }
    }

    if (-f $cfgin or $cfgin eq "&STDIN")
    {

      open CFGIN, "<$cfgin" or die "Error: Cannot open $cfgin.\n";
      print STDERR "Reading from $cfgin...\n";

      if (-f $cfgout)
      {
        warn "Warning: $cfgout allready exists, press ENTER to overwrite or CTRL-C to abort.\n";
        read STDIN, $foo, 1;
      }

      open CFGOUT, ">$cfgout" or die "Error: Cannot open $cfgout for writing.\n";
      print STDERR "Writing to $cfgout...\n";

      $date                                =                                sprintf("%04d-%02d-%02d",
(1900+(localtime)[5]),(1+(localtime)[4]),(1+(localtime)[3]));
      $time = sprintf("%02d:%02d:%02d", ((localtime)[2]),((localtime)[1]),((localtime)[0]));

      print CFGOUT "!\n! Configuration changed by $0 on $date, $time.\n!\n";

      $line_count = 0;
      while ($line = <CFGIN>)
      {
        if ($line =~ /$old_prefix/i)
        {
          if ($mode =~ /add/)
          {
            print CFGOUT $line;
          }
          elsif ($mode =~ /replace/)
          {
            if ($line =~ /^\s/)
            {
              print CFGOUT " no$line";
            }
            else
            {
              print CFGOUT "no $line";
            }
          }
```

```
        if ($mode =~ /interfaceonly/ and $line =~ /ipv6 address/)
        {
          $line =~ s/$old_prefix/$new_prefix/ig;
        }
        else
        {
          $line = "";
        }
      }
      print CFGOUT $line;
      $line_count++;
    }

    print STDERR "Processed $line_count lines. Done.\n";
}

# EOF
```