

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 22, 2016

J. Chroboczek
PPS, University of Paris-Diderot
March 21, 2016

Homenet profile of the Babel routing protocol
draft-chroboczek-homenet-babel-profile-00

Abstract

This document defines the subset of the Babel routing protocol [RFC6126] and its extensions that a Homenet router must implement.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 22, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Background	2
2. The Homenet profile of Babel	3
2.1. Requirements	3
2.2. Non-requirements	5
3. Acknowledgments	6
4. References	6
4.1. Normative References	6
4.2. Informative References	6
Author's Address	6

1. Introduction

The core of the Homenet protocol suite consists of HNCP [HNCP], a protocol used for flooding configuration information and assigning prefixes to links, combined with the Babel routing protocol [RFC6126]. Babel is an extensible, flexible and modular protocol: minimal implementations of Babel have been demonstrated that consist of a few hundred of lines of code, while the "large" implementation includes support for a number of extensions and consists of over ten thousand lines of C code.

This document defines the exact subset of the Babel protocol and its extensions that is required by a conformant implementation of the Homenet protocol suite.

1.1. Background

The Babel routing protocol and its extensions are defined in a number of documents:

- o The body of RFC 6126 [RFC6126] defines the core, unextended protocol. It allows Babel's control data to be carried over either link-local IPv6 or IPv4, and in either case allows announcing both IPv4 and IPv6 routes. It leaves link cost estimation, metric computation and route selection to the implementation. Distinct implementations of core RFC 6126 Babel will interoperate and maintain a set of loop-free forwarding paths, but given conflicting metrics or route selection policies may give rise to persistent oscillations.
- o The informative Appendix A of RFC 6126 suggests a simple and easy to implement algorithm for cost and metric computation that has been found to work satisfactorily in a wide range of topologies.

- o While RFC 6126 does not provide an algorithm for route selection, its Section 3.6 suggests selecting the route with smallest metric with some hysteresis applied. An algorithm that has been found to work well in practice is described in Section III.E of [DELAY-BASED].
- o The extension mechanism for Babel is defined in RFC 7557 [RFC7557].
- o Four RFCs and Internet-Drafts define optional extensions to Babel: HMAC-based authentication [RFC7298], source-specific routing [BABEL-SS], radio interference aware routing [BABEL-Z], and delay-based routing [BABEL-RTT]. All of these extensions interoperate with the core protocol as well as with each other.

2. The Homenet profile of Babel

2.1. Requirements

[Sentences within square brackets are editorial notes and are not intended for publication.]

REQ1: a Homenet implementation of Babel MUST encapsulate Babel control traffic in IPv6 packets sent to the IANA-assigned port 6696 and either the IANA-assigned multicast group ff02::1:6 or to a link-local unicast address.

Rationale: since Babel is able to carry both IPv4 and IPv6 routes over either IPv4 or IPv6, choosing the protocol used for carrying control traffic is a matter of preference. Since IPv6 has some features that make implementations somewhat simpler and more reliable (notably link-local addresses), we require carrying control data over IPv6.

REQ2: a Homenet implementation of Babel MUST implement the IPv6 subset of the protocol defined in the body of RFC 6126.

Rationale: support for IPv6 routing is an essential component of the Homenet architecture.

REQ3: a Homenet implementation of Babel SHOULD implement the IPv4 subset of the protocol defined in the body of RFC 6126. Use of other techniques for acquiring IPv4 connectivity (such as multiple layers of NAT) is strongly discouraged.

Rationale: support for IPv4 will remain necessary for years to come, and even in pure IPv6 deployments, including code for supporting IPv4 has very little cost. Since HNCP makes it easy to

assign distinct IPv4 prefixes to the links in a network, it is not necessary to resort to multiple layers of NAT, with all of its problems.

[BS suggest that this should be a MUST.]

REQ4: a Homenet implementation of Babel MUST implement source-specific routing for IPv6, as defined in draft-boutier-babel-source-specific [BABEL-SS]. This implies that it MUST implement the extension mechanism defined in RFC 7557.

Rationale: source-specific routing is an essential component of the Homenet architecture. The extension mechanism is required by source-specific routing. Source-specific routing for IPv4 is not required, since HNCP arranges things so that a single non-specific IPv4 default route is announced (Section 6.5 of [HNCP]).

REQ5: a Homenet implementation of Babel MUST implement HMAC-based authentication, as defined in RFC 7298, MUST implement the two mandatory-to-implement algorithms defined in RFC 7298, and MUST enable and require authentication when instructed to do so by HNCP.

Rationale: some home networks include "guest" links that can be used by third parties that are not necessarily fully trusted. In such networks, it is essential that either the routing protocol is secured or the guest links are carefully firewalled.

Generic mechanisms such as DTLS and dynamically keyed IPsec are not able to protect multicast traffic, and are therefore difficult to use with Babel. Statically keyed IPsec, perhaps with keys rotated by HNCP, is vulnerable to replay attacks and would therefore require the addition of a nonce mechanism to Babel.

[There is no consensus about this requirement. A simpler solution is to disable Babel on guest interfaces. MS suggests this might be a SHOULD.]

[This needs expanding with an explanation of how HNCP is supposed to signal the use of authentication.]

REQ6: a Homenet implementation of Babel MUST use metrics that are of a similar magnitude to the values suggested in Appendix A of RFC 6126. In particular, it SHOULD assign costs that are no less than 256 to wireless links, and SHOULD assign costs between 32 and 196 to lossless wired links.

Rationale: if two implementations of Babel choose very different values for link costs, combining routers from different vendors will lead to sub-optimal routing.

REQ7: a Homenet implementation of Babel SHOULD distinguish between wired and wireless links; if it is unable to determine whether a link is wired or wireless, it SHOULD make the worst-case hypothesis that the link is wireless. It SHOULD dynamically probe the quality of wireless links and derive a suitable metric from its quality estimation. The algorithm described in Appendix A of RFC 6126 MAY be used.

Rationale: support for wireless transit links is a "killer feature" of Homenet, something that is requested by our users and easy to explain to our bosses. In the absence of dynamically computed metrics, the routing protocol attempts to minimise the number of links crossed by a route, and therefore prefers long, lossy links to shorter, lossless ones. In wireless networks, "hop-count routing is worst-path routing".

[This should probably be MUST, but it might be difficult or even impossible to implement in some environments, especially in the presence of wired-to-wireless bridges.]

2.2. Non-requirements

NR1: a Homenet implementation of Babel MAY perform route selection by applying hysteresis to route metrics, as suggested in Section 3.6 of RFC 6126 and described in detail in Section III.E of [BABEL-RTT]. However, it MAY simply pick the route with the smallest metric.

Rationale: hysteresis is only useful in congested and highly dynamic networks. In a typical home network, stable and uncongested, the feedback loop that hysteresis compensates for does not occur.

NR2: a Homenet implementation of Babel MAY include support for other extensions to the protocol, as long as they are known to interoperate with both the core protocol and source-specific routing.

Rationale: delay-based routing is useful in redundant meshes of tunnels, which do not occur in typical home networks (which typically use at most one VPN link). Interference-aware routing, on the other hand, is likely to be useful in home networks, but the extension requires further evaluation before it can be recommended for widespread deployment.

3. Acknowledgments

4. References

4.1. Normative References

- [BABEL-SS] Boutier, M. and J. Chroboczek, "Source-Specific Routing in Babel", draft-boutier-babel-source-specific-01 (work in progress), January 2015.
- [RFC6126] Chroboczek, J., "The Babel Routing Protocol", RFC 6126, February 2011.
- [RFC7298] Ovsienko, D., "Babel Hashed Message Authentication Code (HMAC) Cryptographic Authentication", RFC 7298, July 2014.
- [RFC7557] Chroboczek, J., "Extension Mechanism for the Babel Routing Protocol", RFC 7557, May 2015.

4.2. Informative References

- [BABEL-RTT] Jonglez, B. and J. Chroboczek, "Delay-based Metric Extension for the Babel Routing Protocol", draft-jonglez-babel-rtt-extension-01 (work in progress), May 2015.
- [BABEL-Z] Chroboczek, J., "Diversity Routing for the Babel Routing Protocol", draft-chroboczek-babel-diversity-routing-00 (work in progress), July 2014.
- [DELAY-BASED] Jonglez, B. and J. Chroboczek, "A delay-based routing metric", March 2014.

Available online from <http://arxiv.org/abs/1403.3488>
- [HNCP] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", draft-ietf-homenet-hncp-09 (work in progress), August 2015.

Author's Address

Juliusz Chroboczek
PPS, University of Paris-Diderot
Case 7014
75205 Paris Cedex 13
France

Email: jch@pps.univ-paris-diderot.fr

Homenet Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2016

M. Stenberg
S. Barth
Independent
November 2, 2015

Distributed Node Consensus Protocol
draft-ietf-homenet-dncp-12

Abstract

This document describes the Distributed Node Consensus Protocol (DNC), a generic state synchronization protocol that uses the Trickle algorithm and hash trees. DNC is an abstract protocol, and must be combined with a specific profile to make a complete implementable protocol.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 5, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Applicability	3
2. Terminology	6
2.1. Requirements Language	8
3. Overview	8
4. Operation	9
4.1. Hash Tree	9
4.1.1. Calculating network state and node data hashes	9
4.1.2. Updating network state and node data hashes	10
4.2. Data Transport	10
4.3. Trickle-Driven Status Updates	11
4.4. Processing of Received TLVs	12
4.5. Discovering, Adding and Removing Peers	15
4.6. Data Liveliness Validation	16
5. Data Model	17
6. Optional Extensions	19
6.1. Keep-Alives	19
6.1.1. Data Model Additions	19
6.1.2. Per-Endpoint Periodic Keep-Alives	20
6.1.3. Per-Peer Periodic Keep-Alives	20
6.1.4. Received TLV Processing Additions	20
6.1.5. Peer Removal	20
6.2. Support For Dense Multicast-Enabled Links	21
7. Type-Length-Value Objects	22
7.1. Request TLVs	23
7.1.1. Request Network State TLV	23
7.1.2. Request Node State TLV	23
7.2. Data TLVs	23
7.2.1. Node Endpoint TLV	23
7.2.2. Network State TLV	24
7.2.3. Node State TLV	24
7.3. Data TLVs within Node State TLV	25
7.3.1. Peer TLV	25
7.3.2. Keep-Alive Interval TLV	26
8. Security and Trust Management	26
8.1. Pre-Shared Key Based Trust Method	26
8.2. PKI Based Trust Method	27
8.3. Certificate Based Trust Consensus Method	27
8.3.1. Trust Verdicts	27
8.3.2. Trust Cache	28
8.3.3. Announcement of Verdicts	29
8.3.4. Bootstrap Ceremonies	30
9. DNPCP Profile-Specific Definitions	31
10. Security Considerations	33
11. IANA Considerations	33
12. References	34

12.1. Normative references	34
12.2. Informative references	34
Appendix A. Alternative Modes of Operation	35
A.1. Read-only Operation	35
A.2. Forwarding Operation	35
Appendix B. DNCP Profile Additional Guidance	36
B.1. Unicast Transport - UDP or TCP?	36
B.2. (Optional) Multicast Transport	36
B.3. (Optional) Transport Security	37
Appendix C. Example Profile	37
Appendix D. Some Questions and Answers [RFC Editor: please remove]	38
Appendix E. Changelog [RFC Editor: please remove]	38
Appendix F. Draft Source [RFC Editor: please remove]	40
Appendix G. Acknowledgements	40
Authors' Addresses	41

1. Introduction

DNCP is designed to provide a way for each participating node to publish a small set of TLV (Type-Length-Value) tuples (at most 64 KB), and to provide a shared and common view about the data published by every currently bidirectionally reachable DNCP node in a network.

For state synchronization a hash tree is used. It is formed by first calculating a hash for the dataset published by each node, called node data, and then calculating another hash over those node data hashes. The single resulting hash, called network state hash, is transmitted using the Trickle algorithm [RFC6206] to ensure that all nodes share the same view of the current state of the published data within the network. The use of Trickle with only short network state hashes sent infrequently (in steady state, once the maximum Trickle interval per link or unicast connection has been reached) makes DNCP very thrifty when updates happen rarely.

For maintaining liveliness of the topology and the data within it, a combination of Trickled network state, keep-alives, and "other" means of ensuring reachability are used. The core idea is that if every node ensures its peers are present, transitively, the whole network state also stays up-to-date.

1.1. Applicability

DNCP is useful for cases like autonomous bootstrapping, discovery and negotiation of embedded network devices like routers. Furthermore it can be used as a basis to run distributed algorithms like [I-D.ietf-homenet-prefix-assignment] or usecases as described in Appendix C. DNCP is abstract, which allows it to be tuned to a

variety of applications by defining profiles. These profiles include choices of:

- unicast transport: datagram or stream oriented protocol (e.g., TCP, UDP, SCTP) for generic protocol operation
- optional transport security: whether and when to use security based on (D)TLS, if supported over the chosen transport
- optional multicast transport: multicast-capable protocol like UDP allowing autonomous peer discovery or more efficient use of multiple access links
- communication scopes: either hop-by-hop only relying on link-local addressing (e.g., for LANs) or using addresses with broader scopes (e.g. over WANs or the internet) relying on an existing routing infrastructure or a combination of both (e.g., to exchange state between multiple LANs over a WAN or the internet)
- payloads: additional specific payloads (e.g., IANA standardized, enterprise-specific or private use)
- extensions: possible protocol extensions, either as predefined in this document or specific for a particular usecase

However, there are certain cases where the protocol as defined in this document is a less suitable choice. This list provides an overview while the following paragraphs provide more detailed guidance on the individual matters.

- large amounts of data: nodes are limited to 64KB of published data
- very dense unicast-only networks: nodes include information about all immediate neighbors as part of their published data.
- predominantly minimal data changes: Node data is always transported as-is, leading to a relatively large transmission overhead for changes affecting only a small part of it.
- frequently changing data: DNCP with its use of Trickle is optimized for the steady state and less efficient otherwise.
- large amounts of very constrained nodes: DNCP requires each node to store the entirety of the data published by all nodes.

The topology of the devices is not limited and automatically discovered. When relying on link-local communication exclusively, all links having DNCP nodes need to be at least transitively

connected by routers running the protocol on multiple endpoints in order to form a connected network. However, there is no requirement for every device in a physical network to run the protocol. Especially if globally scoped addresses are used, DNCP peers do not need to be on the same or even neighboring physical links. Autonomous discovery features are usually used in local network scenario however - with security enabled - DNCP can also be used over unsecured public networks. Network size is restricted merely by the capabilities of the devices, i.e., each DNCP node needs to be able to store the entirety of the data published by all nodes. The data associated with each individual node identifier is limited to about 64KB in this document, however protocol extensions could be defined to mitigate this or other protocol limitations if the need arises.

DNCP is most suitable for data that changes only infrequently to gain the maximum benefit from using Trickle. As the network of nodes grows, or the frequency of data changes per node increases, Trickle is eventually used less and less and the benefit of using DNCP diminishes. In these cases Trickle just provides extra complexity within the specification and little added value.

The suitability of DNCP for a particular application can roughly be evaluated by considering the expected average network-wide state change interval A_{NC_I} ; it is computed by dividing the mean interval at which a node originates a new TLV set by the number of participating nodes. If keep-alives are used, A_{NC_I} is the minimum of the computed A_{NC_I} and the keep-alive interval. If A_{NC_I} is less than the (application-specific) Trickle minimum interval, DNCP is most likely unsuitable for the application as Trickle will not be utilized most of the time.

If constant rapid state changes are needed, the preferable choice is to use an additional point-to-point channel whose address or locator is published using DNCP. Nevertheless, if doing so does not raise A_{NC_I} above the (sensibly chosen) Trickle interval parameters for a particular application, using DNCP is probably not suitable for the application.

Another consideration is the size of the published TLV set by a node compared to the size of deltas in the TLV set. If the TLV set published by a node is very large, and has frequent small changes, DNCP as currently specified in this specification may be unsuitable as it lacks a delta synchronization scheme to keep implementation simple.

DNCP can be used in networks where only unicast transport is available. While DNCP uses the least amount of bandwidth when multicast is utilized, even in pure unicast mode, the use of Trickle

(ideally with $k < 2$) results in a protocol with an exponential backoff timer and fewer transmissions than a simpler protocol not using Trickle.

2. Terminology

DNCP profile	the values for the set of parameters, given in Section 9. They are prefixed with DNCP_ in this document. The profile also specifies the set of optional DNCP extensions to be used. For a simple example DNCP profile, see Appendix C.
DNCP-based protocol	a protocol which provides a DNCP profile, according to Section 9, and zero or more TLV assignments from the per-DNCP profile TLV registry as well as their processing rules.
DNCP node	a single node which runs a DNCP-based protocol.
Link	a link-layer media over which directly connected nodes can communicate.
DNCP network	a set of DNCP nodes running DNCP-based protocol(s) with matching DNCP profile(s). The set consists of nodes that have discovered each other using the transport method defined in the DNCP profile, via multicast on local links, and / or by using unicast communication.
Node identifier	an opaque fixed-length identifier consisting of DNCP_NODE_IDENTIFIER_LENGTH bytes which uniquely identifies a DNCP node within a DNCP network.
Interface	a node's attachment to a particular link.
Address	an identifier used as source or destination of a DNCP message flow, e.g., a tuple (IPv6 address, UDP port) for an IPv6 UDP transport.
Endpoint	a locally configured termination point for (potential or established) DNCP message flows. An endpoint is the source and destination for separate unicast message flows to individual nodes and optionally for multicast messages to all thereby reachable nodes (e.g., for node discovery). Endpoints are usually in one of the transport modes specified in Section 4.2.

Endpoint identifier	a 32-bit opaque and locally unique value, which identifies a particular endpoint of a particular DNCP node. The value 0 is reserved for DNCP and DNCP-based protocol purposes and not used to identify an actual endpoint. This definition is in sync with the interface index definition in [RFC3493], as the non-zero small positive integers should comfortably fit within 32 bits.
Peer	another DNCP node with which a DNCP node communicates using at least one particular local and remote endpoint pair.
Node data	a set of TLVs published and owned by a node in the DNCP network. Other nodes pass it along as-is, even if they cannot fully interpret it.
Origination Time	the (estimated) time when the node data set with the current sequence number was published.
Node state	a set of metadata attributes for node data. It includes a sequence number for versioning, a hash value for comparing equality of stored node data, and a timestamp indicating the time passed since its last publication (i.e., since the origination time). The hash function and the length of the hash value are defined in the DNCP profile.
Network state hash	a hash value which represents the current state of the network. The hash function and the length of the hash value are defined in the DNCP profile. Whenever a node is added, removed or updates its published node data this hash value changes as well. For calculation, please see Section 4.1.
Trust verdict	a statement about the trustworthiness of a certificate announced by a node participating in the certificate based trust consensus mechanism.
Effective trust verdict	the trust verdict with the highest priority within the set of trust verdicts announced for the certificate in the DNCP network.
Topology graph	the undirected graph of DNCP nodes produced by retaining only bidirectional peer relationships between nodes.
Bidirectionally	a peer is locally unidirectionally reachable if a

reachable consistent multicast or any unicast DNCP message has been received by the local node (see Section 4.5). If said peer in return also considers the local node unidirectionally reachable, then bidirectionally reachability is established. As this process is based on publishing peer relationships and evaluating the resulting topology graph as described in Section 4.6, this information is available to the whole DNCP network.

Trickle Instance a distinct Trickle [RFC6206] algorithm state kept by a node (Section 5) and related to an endpoint or a particular (peer, endpoint) tuple with Trickle variables *I*, *t* and *c*. See Section 4.3.

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Overview

DNCP operates primarily using unicast exchanges between nodes, and may use multicast for Trickle-based shared state dissemination and topology discovery. If used in pure unicast mode with unreliable transport, Trickle is also used between peers.

DNCP is based on exchanging TLVs (Section 7) and defines a set of mandatory and optional ones for its operation. They are categorized into TLVs for requesting information (Section 7.1), transmitting data (Section 7.2) and being published as data (Section 7.3). DNCP based protocols usually specify additional ones to extend the capabilities.

DNCP discovers the topology of the nodes in the DNCP network and maintains the liveliness of published node data by ensuring that the publishing node is bidirectionally reachable. New potential peers can be discovered autonomously on multicast-enabled links, their addresses may be manually configured or they may be found by some other means defined in the particular DNCP profile. The DNCP profile may specify, for example, a well-known anycast address or provisioning the remote address to contact via some other protocol such as DHCPv6 [RFC3315].

A hash tree of height 1, rooted in itself, is maintained by each node to represent the state of all currently reachable nodes (see Section 4.1) and the Trickle algorithm is used to trigger

synchronization (see Section 4.3). The need to check peer nodes for state changes is thereby determined by comparing the current root of their respective hash trees, i.e., their individually calculated network state hashes.

Before joining a DNCP network, a node starts with a hash tree that has only one leaf if the node publishes some TLVs, and no leaves otherwise. It then announces the network state hash calculated from the hash tree by means of the Trickle algorithm on all its configured endpoints.

When an update is detected by a node (e.g., by receiving a different network state hash from a peer) the originator of the event is requested to provide a list of the state of all nodes, i.e., all the information it uses to calculate its own hash tree. The node uses the list to determine whether its own information is outdated and - if necessary - requests the actual node data that has changed.

Whenever a node's local copy of any node data and its hash tree are updated (e.g., due to its own or another node's node state changing or due to a peer being added or removed) its Trickle instances are reset which eventually causes any update to be propagated to all of its peers.

4. Operation

4.1. Hash Tree

Each DNCP node maintains an arbitrary width hash tree of height 1. The root of the tree represents the overall network state hash and is used to determine whether the view of the network of two or more nodes is consistent and shared. Each leaf represents one bidirectionally reachable DNCP node. Every time a node is added or removed from the topology graph (Section 4.6) it is likewise added or removed as a leaf. At any time the leaves of the tree are ordered in ascending order of the node identifiers of the nodes they represent.

4.1.1. Calculating network state and node data hashes

The network state hash and the node data hashes are calculated using the hash function defined in the DNCP profile (Section 9) and truncated to the number of bits specified therein.

Individual node data hashes are calculated by applying the function and truncation on the respective node's node data as published in the Node State TLV. Such node data sets are always ordered as defined in Section 7.2.3.

The network state hash is calculated by applying the function and truncation on the concatenated network state. This state is formed by first concatenating each node's sequence number (in network byte order) with its node data hash to form a per-node datum for each node. These per-node data are then concatenated in ascending order of the respective node's node identifier, i.e., in the order that the nodes appear in the hash tree.

4.1.2. Updating network state and node data hashes

The network state hash and the node data hashes are updated on-demand and whenever any locally stored per-node state changes. This includes local unidirectional reachability encoded in the published Peer TLVs (Section 7.3.1) and - when combined with remote data - results in awareness of bidirectional reachability changes.

4.2. Data Transport

DNCP has few requirements for the underlying transport; it requires some way of transmitting either unicast datagram or stream data to a peer and, if used in multicast mode, a way of sending multicast datagrams. As multicast is used only to identify potential new DNCP nodes and to send status messages which merely notify that a unicast exchange should be triggered, the multicast transport does not have to be secured. If unicast security is desired and one of the built-in security methods is to be used, support for some TLS-derived transport scheme - such as TLS [RFC5246] on top of TCP or DTLS [RFC6347] on top of UDP - is also required. They provide for integrity protection and confidentiality of the node data, as well as authentication and authorization using the schemes defined in Security and Trust Management (Section 8). A specific definition of the transport(s) in use and their parameters **MUST** be provided by the DNCP profile.

TLVs (Section 7) are sent across the transport as is, and they **SHOULD** be sent together where, e.g., MTU considerations do not recommend sending them in multiple batches. DNCP does not fragment or reassemble TLVs thus it **MUST** be ensured that the underlying transport performs these operations should they be necessary. If this document indicates sending one or more TLVs, then the sending node does not need to keep track of the packets sent after handing them over to the respective transport, i.e., reliable DNCP operation is ensured merely by the explicitly defined timers and state machines such as Trickle (Section 4.3). TLVs in general are handled individually and statelessly (and thus do not need to be sent in any particular order) with one exception: To form bidirectional peer relationships DNCP requires identification of the endpoints used for communication. As bidirectional peer relationships are required for validating

liveliness of published node data as described in Section 4.6, a DNCP node MUST send a Node Endpoint TLV (Section 7.2.1). When it is sent varies, depending on the underlying transport, but conceptually it should be available whenever processing a Network State TLV:

- o If using a stream transport, the TLV MUST be sent at least once per connection, but SHOULD NOT be sent more than once.
- o If using a datagram transport, it MUST be included in every datagram that also contains a Network State TLV (Section 7.2.2) and MUST be located before any such TLV. It SHOULD also be included in any other datagram, to speed up initial peer detection.

Given the assorted transport options as well as potential endpoint configuration, a DNCP endpoint may be used in various transport modes:

Unicast:

- * If only reliable unicast transport is used, Trickle is not used at all. Whenever the locally calculated network state hash changes, a single Network State TLV (Section 7.2.2) is sent to every unicast peer. Additionally, recently changed Node State TLVs (Section 7.2.3) MAY be included.
- * If only unreliable unicast transport is used, Trickle state is kept per peer and it is used to send Network State TLVs intermittently, as specified in Section 4.3.

Multicast+Unicast: If multicast datagram transport is available on an endpoint, Trickle state is only maintained for the endpoint as a whole. It is used to send Network State TLVs periodically, as specified in Section 4.3. Additionally, per-endpoint keep-alives MAY be defined in the DNCP profile, as specified in Section 6.1.2.

MulticastListen+Unicast: Just like Unicast, except multicast transmissions are listened to in order to detect changes of the highest node identifier. This mode is used only if the DNCP profile supports dense multicast-enabled link optimization (Section 6.2).

4.3. Trickle-Driven Status Updates

The Trickle algorithm [RFC6206] is used to ensure protocol reliability over unreliable multicast or unicast transports. For reliable unicast transports, its actual algorithm is unnecessary and omitted (Section 4.2). DNCP maintains multiple Trickle states as

defined in Section 5. Each such state can be based on different parameters (see below) and is responsible for ensuring that a specific peer or all peers on the respective endpoint are regularly provided with the node's current locally calculated network state hash for state comparison, i.e., to detect potential divergence in the perceived network state.

Trickle defines 3 parameters: Imin, Imax and k. Imin and Imax represent the minimum value for I and the maximum number of doublings of Imin, where I is the time interval during which at least k Trickle updates must be seen on an endpoint to prevent local state transmission. The actual suggested Trickle algorithm parameters are DNCP profile specific, as described in Section 9.

The Trickle state for all Trickle instances defined in Section 5 is considered inconsistent and reset if and only if the locally calculated network state hash changes. This occurs either due to a change in the local node's own node data, or due to receipt of more recent data from another node as explained in Section 4.1. A node MUST NOT reset its Trickle state merely based on receiving a Network State TLV (Section 7.2.2) with a network state hash which is different from its locally calculated one.

Every time a particular Trickle instance indicates that an update should be sent, the node MUST send a Network State TLV (Section 7.2.2) if and only if:

- o the endpoint is in Multicast+Unicast transport mode, in which case the TLV MUST be sent over multicast.
- o the endpoint is NOT in Multicast+Unicast transport mode, and the unicast transport is unreliable, in which case the TLV MUST be sent over unicast.

A (sub)set of all Node State TLVs (Section 7.2.3) MAY also be included, unless it is defined as undesirable for some reason by the DNCP profile, or to avoid exposure of the node state TLVs by transmitting them within insecure multicast when using secure unicast.

4.4. Processing of Received TLVs

This section describes how received TLVs are processed. The DNCP profile may specify when to ignore particular TLVs, e.g., to modify security properties - see Section 9 for what may be safely defined to be ignored in a profile. Any 'reply' mentioned in the steps below denotes sending of the specified TLV(s) to the originator of the TLV being processed. All such replies MUST be sent using unicast. If

the TLV being replied to was received via multicast and it was sent to a multiple access link, the reply MUST be delayed by a random timespan in $[0, I_{min}/2]$, to avoid potential simultaneous replies that may cause problems on some links, unless specified differently in the DNCP profile. Sending of replies MAY also be rate-limited or omitted for a short period of time by an implementation. However, if the TLV is not forbidden by the DNCP profile, an implementation MUST reply to retransmissions of the TLV with a non-zero probability to avoid starvation which would break the state synchronization.

A DNCP node MUST process TLVs received from any valid (e.g., correctly scoped) address, as specified by the DNCP profile and the configuration of a particular endpoint, whether this address is known to be the address of a peer or not. This provision satisfies the needs of monitoring or other host software that needs to discover the DNCP topology without adding to the state in the network.

Upon receipt of:

- o Request Network State TLV (Section 7.1.1): The receiver MUST reply with a Network State TLV (Section 7.2.2) and a Node State TLV (Section 7.2.3) for each node data used to calculate the network state hash. The Node State TLVs SHOULD NOT contain the optional node data part to avoid redundant transmission of node data, unless explicitly specified in the DNCP profile.
- o Request Node State TLV (Section 7.1.2): If the receiver has node data for the corresponding node, it MUST reply with a Node State TLV (Section 7.2.3) for the corresponding node. The optional node data part MUST be included in the TLV.
- o Network State TLV (Section 7.2.2): If the network state hash differs from the locally calculated network state hash, and the receiver is unaware of any particular node state differences with the sender, the receiver MUST reply with a Request Network State TLV (Section 7.1.1). These replies MUST be rate limited to only at most one reply per link per unique network state hash within I_{min} . The simplest way to ensure this rate limit is a timestamp indicating requests, and sending at most one Request Network State TLV (Section 7.1.1) per I_{min} . To facilitate faster state synchronization, if a Request Network State TLV is sent in a reply, a local, current Network State TLV MAY also be sent.
- o Node State TLV (Section 7.2.3):
 - * If the node identifier matches the local node identifier and the TLV has a greater sequence number than its current local value, or the same sequence number and a different hash, the

node SHOULD re-publish its own node data with a sequence number significantly (e.g., 1000) greater than the received one, to reclaim the node identifier. This difference is needed in order to ensure that it is higher than any potentially lingering copies of the node state in the network. This may occur normally once due to the local node restarting and not storing the most recently used sequence number. If this occurs more than once or for nodes not re-publishing their own node data, the DNCP profile MUST provide guidance on how to handle these situations as it indicates the existence of another active node with the same node identifier.

- * If the node identifier does not match the local node identifier, and one or more of the following conditions are true:
 - + The local information is outdated for the corresponding node (local sequence number is less than that within the TLV).
 - + The local information is potentially incorrect (local sequence number matches but the node data hash differs).
 - + There is no data for that node altogether.

Then:

- + If the TLV contains the Node Data field, it SHOULD also be verified by ensuring that the locally calculated hash of the Node Data matches the content of the H(Node Data) field within the TLV. If they differ, the TLV SHOULD be ignored and not processed further.
- + If the TLV does not contain the Node Data field, and the H(Node Data) field within the TLV differs from the local node data hash for that node (or there is none), the receiver MUST reply with a Request Node State TLV (Section 7.1.2) for the corresponding node.
- + Otherwise the receiver MUST update its locally stored state for that node (node data based on Node Data field if present, sequence number and relative time) to match the received TLV.

For comparison purposes of the sequence number, a looping comparison function MUST be used to avoid problems in case of overflow. The comparison function $a < b \Leftrightarrow ((a - b) \% (2^{32})) \& (2^{31}) \neq 0$ where $(a \% b)$ represents the remainder of a modulo b

and (a & b) represents bitwise conjunction of a and b is RECOMMENDED unless the DNCP profile defines another.

- o Any other TLV: TLVs not recognized by the receiver MUST be silently ignored unless they are sent within another TLV (for example, TLVs within the Node Data field of a Node State TLV). TLVs within the Node Data field of the Node State TLV not recognized by the receiver MUST be retained for distribution to other nodes and for calculating the node data hash as described in Section 7.2.3 but are ignored for other purposes.

If secure unicast transport is configured for an endpoint, any Node State TLVs received over insecure multicast MUST be silently ignored.

4.5. Discovering, Adding and Removing Peers

Peer relations are established between neighbors using one or more mutually connected endpoints. Such neighbors exchange information about network state and published data directly and through transitivity this information then propagates throughout the network.

New peers are discovered using the regular unicast or multicast transport defined in the DNCP profile (Section 9). This process is not distinguished from peer addition, i.e., an unknown peer is simply discovered by receiving regular DNCP protocol TLVs from it and dedicated discovery messages or TLVs do not exist. For unicast-only transports, the individual node's transport addresses are preconfigured or obtained using an external service discovery protocol. In the presence of a multicast transport, messages from unknown peers are handled in the same way as multicast messages from peers that are already known, thus new peers are simply discovered when sending their regular DNCP protocol TLVs using multicast.

When receiving a Node Endpoint TLV (Section 7.2.1) on an endpoint from an unknown peer:

- o If received over unicast, the remote node MUST be added as a peer on the endpoint and a Peer TLV (Section 7.3.1) MUST be created for it.
- o If received over multicast, the node MAY be sent a (possibly rate-limited) unicast Request Network State TLV (Section 7.1.1).

If keep-alives specified in Section 6.1 are NOT sent by the peer (either the DNCP profile does not specify the use of keep-alives or the particular peer chooses not to send keep-alives), some other existing local transport-specific means (such as Ethernet carrier-detection or TCP keep-alive) MUST be used to ensure its presence. If

the peer does not send keep-alives, and no means to verify presence of the peer are available, the peer MUST be considered no longer present and it SHOULD NOT be added back as a peer until it starts sending keep-alives again. When the peer is no longer present, the Peer TLV and the local DNCP peer state MUST be removed. DNCP does not define an explicit message or TLV for indicating the termination of DNCP operation by the terminating node, however a derived protocol could specify an extension, if the need arises.

If the local endpoint is in the Multicast-Listen+Unicast transport mode, a Peer TLV (Section 7.3.1) MUST NOT be published for the peers not having the highest node identifier.

4.6. Data Liveliness Validation

Maintenance of the hash tree (Section 4.1) and thereby network state hash updates depend on up-to-date information on bidirectional node reachability derived from the contents of a topology graph. This graph changes whenever nodes are added to or removed from the network or when bidirectional connectivity between existing nodes is established or lost. Therefore the graph MUST be updated either immediately or with a small delay shorter than the DNCP profile-defined Trickle Imin, whenever:

- o A Peer TLV or a whole node is added or removed, or
- o the origination time (in milliseconds) of some node's node data is less than current time - $2^{32} + 2^{15}$.

The artificial upper limit for the origination time is used to gracefully avoid overflows of the origination time and allow for the node to republish its data as noted in Section 7.2.3.

The topology graph update starts with the local node marked as reachable and all other nodes marked as unreachable. Other nodes are then iteratively marked as reachable using the following algorithm: A candidate not-yet-reachable node N with an endpoint NE is marked as reachable if there is a reachable node R with an endpoint RE that meet all of the following criteria:

- o The origination time (in milliseconds) of R's node data is greater than current time - $2^{32} + 2^{15}$.
- o R publishes a Peer TLV with:
 - * Peer Node Identifier = N's node identifier
 - * Peer Endpoint Identifier = NE's endpoint identifier

- * Endpoint Identifier = RE's endpoint identifier
- o N publishes a Peer TLV with:
 - * Peer Node Identifier = R's node identifier
 - * Peer Endpoint Identifier = RE's endpoint identifier
 - * Endpoint Identifier = NE's endpoint identifier

The algorithm terminates, when no more candidate nodes fulfilling these criteria can be found.

DNCP nodes that have not been reachable in the most recent topology graph traversal MUST NOT be used for calculation of the network state hash, be provided to any applications that need to use the whole TLV graph, or be provided to remote nodes. They MAY be forgotten immediately after the topology graph traversal, however it is RECOMMENDED to keep them at least briefly to improve the speed of DNCP network state convergence. This reduces the number of queries needed to reconverge during both initial network convergence and when a part of the network loses and regains bidirectional connectivity within that time period.

5. Data Model

This section describes the local data structures a minimal implementation might use. This section is provided only as a convenience for the implementor. Some of the optional extensions (Section 6) describe additional data requirements, and some optional parts of the core protocol may also require more.

A DNCP node has:

- o A data structure containing data about the most recently sent Request Network State TLVs (Section 7.1.1). The simplest option is keeping a timestamp of the most recent request (required to fulfill reply rate limiting specified in Section 4.4).

A DNCP node has for every DNCP node in the DNCP network:

- o Node identifier: the unique identifier of the node. The length, how it is produced, and how collisions are handled, is up to the DNCP profile.
- o Node data: the set of TLV tuples published by that particular node. As they are transmitted ordered (see Node State TLV

(Section 7.2.3) for details), maintaining the order within the data structure here may be reasonable.

- o Latest sequence number: the 32-bit sequence number that is incremented any time the TLV set is published. The comparison function used to compare them is described in Section 4.4.
- o Origination time: the (estimated) time when the current TLV set with the current sequence number was published. It is used to populate the Milliseconds Since Origination field in a Node State TLV (Section 7.2.3). Ideally it also has millisecond accuracy.

Additionally, a DNCP node has a set of endpoints for which DNCP is configured to be used. For each such endpoint, a node has:

- o Endpoint identifier: the 32-bit opaque locally unique value identifying the endpoint within a node. It SHOULD NOT be reused immediately after an endpoint is disabled.
- o Trickle instance: the endpoint's Trickle instance with parameters I, T, and c (only on an endpoint in Multicast+Unicast transport mode).

and one (or more) of the following:

- o Interface: the assigned local network interface.
- o Unicast address: the DNCP node it should connect with.
- o Set of addresses: the DNCP nodes from which connections are accepted.

For each remote (peer, endpoint) pair detected on a local endpoint, a DNCP node has:

- o Node identifier: the unique identifier of the peer.
- o Endpoint identifier: the unique endpoint identifier used by the peer.
- o Peer address: the most recently used address of the peer (authenticated and authorized, if security is enabled).
- o Trickle instance: the particular peer's Trickle instance with parameters I, T, and c (only on an endpoint in Unicast mode, when using an unreliable unicast transport) .

6. Optional Extensions

This section specifies extensions to the core protocol that a DNCP profile may specify to be used.

6.1. Keep-Alives

While DNCP provides mechanisms for discovery and adding of new peers on an endpoint (Section 4.5), as well as state change notifications, another mechanism may be needed to get rid of old, no longer valid peers if the transport or lower layers do not provide one as noted in Section 4.6.

If keep-alives are not specified in the DNCP profile, the rest of this subsection MUST be ignored.

A DNCP profile MAY specify either per-endpoint (sent using multicast to all DNCP nodes connected to a multicast-enabled link) or per-peer (sent using unicast to each peer individually) keep-alive support.

For every endpoint that a keep-alive is specified for in the DNCP profile, the endpoint-specific keep-alive interval MUST be maintained. By default, it is DNCP_KEEPA_LIVE_INTERVAL. If there is a local value that is preferred for that for any reason (configuration, energy conservation, media type, ..), it can be substituted instead. If a non-default keep-alive interval is used on any endpoint, a DNCP node MUST publish appropriate Keep-Alive Interval TLV(s) (Section 7.3.2) within its node data.

6.1.1. Data Model Additions

The following additions to the Data Model (Section 5) are needed to support keep-alives:

For each configured endpoint that has per-endpoint keep-alives enabled:

- o Last sent: If a timestamp which indicates the last time a Network State TLV (Section 7.2.2) was sent over that interface.

For each remote (peer, endpoint) pair detected on a local endpoint, a DNCP node has:

- o Last contact timestamp: a timestamp which indicates the last time a consistent Network State TLV (Section 7.2.2) was received from the peer over multicast, or anything was received over unicast. Failing to update it for a certain amount of time as specified in

Section 6.1.5 results in the removal of the peer. When adding a new peer, it is initialized to the current time.

- o Last sent: If per-peer keep-alives are enabled, a timestamp which indicates the last time a Network State TLV (Section 7.2.2) was sent to to that point-to-point peer. When adding a new peer, it is initialized to the current time.

6.1.2. Per-Endpoint Periodic Keep-Alives

If per-endpoint keep-alives are enabled on an endpoint in Multicast+Unicast transport mode, and if no traffic containing a Network State TLV (Section 7.2.2) has been sent to a particular endpoint within the endpoint-specific keep-alive interval, a Network State TLV (Section 7.2.2) MUST be sent on that endpoint, and a new Trickle interval started, as specified in the step 2 of Section 4.2 of [RFC6206]. The actual sending time SHOULD be further delayed by a random timespan in $[0, I_{min}/2]$.

6.1.3. Per-Peer Periodic Keep-Alives

If per-peer keep-alives are enabled on a unicast-only endpoint, and if no traffic containing a Network State TLV (Section 7.2.2) has been sent to a particular peer within the endpoint-specific keep-alive interval, a Network State TLV (Section 7.2.2) MUST be sent to the peer, and a new Trickle interval started, as specified in the step 2 of Section 4.2 of [RFC6206].

6.1.4. Received TLV Processing Additions

If a TLV is received over unicast from the peer, the Last contact timestamp for the peer MUST be updated.

On receipt of a Network State TLV (Section 7.2.2) which is consistent with the locally calculated network state hash, the Last contact timestamp for the peer MUST be updated in order to maintain it as a peer.

6.1.5. Peer Removal

For every peer on every endpoint, the endpoint-specific keep-alive interval must be calculated by looking for Keep-Alive Interval TLVs (Section 7.3.2) published by the node, and if none exist, using the default value of `DNCP_KEEPALIVE_INTERVAL`. If the peer's Last contact timestamp has not been updated for at least locally chosen potentially endpoint-specific keep-alive multiplier (defaults to `DNCP_KEEPALIVE_MULTIPLIER`) times the peer's endpoint-specific keep-

alive interval, the Peer TLV for that peer and the local DNCP peer state MUST be removed.

6.2. Support For Dense Multicast-Enabled Links

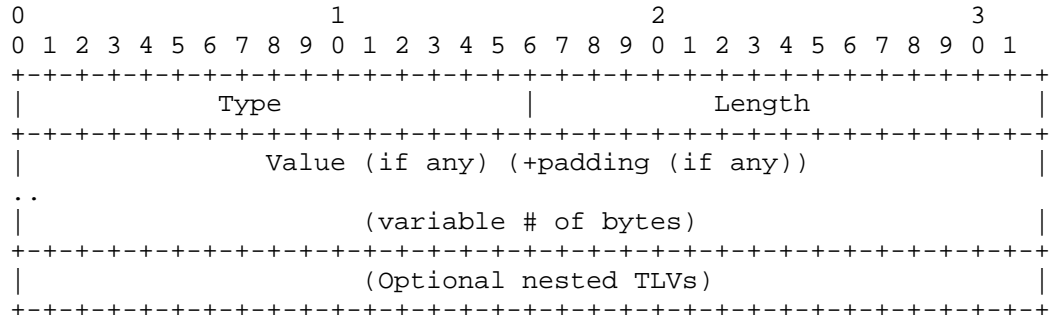
This optimization is needed to avoid a state space explosion. Given a large set of DNCP nodes publishing data on an endpoint that uses multicast on a link, every node will add a Peer TLV (Section 7.3.1) for each peer. While Trickle limits the amount of traffic on the link in stable state to some extent, the total amount of data that is added to and maintained in the DNCP network given N nodes on a multicast-enabled link is $O(N^2)$. Additionally if per-peer keep-alives are used, there will be $O(N^2)$ keep-alives running on the link if liveness of peers is not ensured using some other way (e.g., TCP connection lifetime, layer 2 notification, per-endpoint keep-alive).

An upper bound for the number of peers that are allowed for a particular type of link that an endpoint in Multicast+Unicast transport mode is used on SHOULD be provided by a DNCP profile, but MAY also be chosen at runtime. The main consideration when selecting a bound (if any) for a particular type of link should be whether it supports multicast traffic, and whether a too large number of peers case is likely to happen during the use of that DNCP profile on that particular type of link. If neither is likely, there is little point specifying support for this for that particular link type.

If a DNCP profile does not support this extension at all, the rest of this subsection MUST be ignored. This is because when this extension is used, the state within the DNCP network only contains a subset of the full topology of the network. Therefore every node must be aware of the potential of it being used in a particular DNCP profile.

If the specified upper bound is exceeded for some endpoint in Multicast+Unicast transport mode and if the node does not have the highest node identifier on the link, it SHOULD treat the endpoint as a unicast endpoint connected to the node that has the highest node identifier detected on the link, therefore transitioning to Multicast-listen+Unicast transport mode. See Section 4.2 for implications on the specific endpoint behavior. The nodes in Multicast-listen+Unicast transport mode MUST keep listening to multicast traffic to both receive messages from the node(s) still in Multicast+Unicast mode, and as well to react to nodes with a greater node identifier appearing. If the highest node identifier present on the link changes, the remote unicast address of the endpoints in Multicast-Listen+Unicast transport mode MUST be changed. If the node identifier of the local node is the highest one, the node MUST switch back to, or stay in Multicast+Unicast mode, and form peer relationships with all peers as specified in Section 4.5.

7. Type-Length-Value Objects



Each TLV is encoded as:

- o a 2 byte Type field
- o a 2 byte Length field which contains the length of the Value field in bytes; 0 means no Value
- o the Value itself (if any)
- o padding bytes with value of zero up to the next 4 byte boundary if the Length is not divisible by 4.

While padding bytes MUST NOT be included in the number stored in the Length field of the TLV, if the TLV is enclosed within another TLV, then the padding is included in the enclosing TLV's Length value.

Each TLV which does not define optional fields or variable-length content MAY be sent with additional sub-TLVs appended after the TLV to allow for extensibility. When handling such TLV types, each node MUST accept received TLVs that are longer than the fixed fields specified for the particular type, and ignore the sub-TLVs with either unknown types, or not supported within that particular TLV type. If any sub-TLVs are present, the Length field of the TLV describes the number of bytes from the first byte of the TLV's own Value (if any) to the last (padding) byte of the last sub-TLV.

For example, type=123 (0x7b) TLV with value 'x' (120 = 0x78) is encoded as: 007B 0001 7800 0000. If it were to have sub-TLV of type=124 (0x7c) with value 'y', it would be encoded as 007B 000C 7800 0000 007C 0001 7900 0000.

In this section, the following special notation is used:

.. = octet string concatenation operation.

$H(x)$ = non-cryptographic hash function specified by DNCP profile.

7.1. Request TLVs

7.1.1. Request Network State TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  Type: REQ-NETWORK-STATE (1)  |           Length: >= 0           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

This TLV is used to request response with a Network State TLV (Section 7.2.2) and all Node State TLVs (Section 7.2.3) (without node data).

7.1.2. Request Node State TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  Type: REQ-NODE-STATE (2)  |           Length: > 0           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Node Identifier                    |
|                               (length fixed in DNCP profile)      |
|                               ...                                  |
|                               ...                                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

This TLV is used to request a Node State TLV (Section 7.2.3) (including node data) for the node with the matching node identifier.

7.2. Data TLVs

7.2.1. Node Endpoint TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  Type: NODE-ENDPOINT (3)  |           Length: > 4           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Node Identifier                    |
|                               (length fixed in DNCP profile)      |
|                               ...                                  |
|                               ...                                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Endpoint Identifier                |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

This TLV identifies both the local node's node identifier, as well as the particular endpoint's endpoint identifier. Section 4.2 specifies when it is sent.

7.2.2. Network State TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type: NETWORK-STATE (4)   |   Length: > 0   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   H(sequence number of node 1 .. H(node data of node 1) ..
|   .. sequence number of node N .. H(node data of node N))
|   (length fixed in DNCP profile)
|
...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

This TLV contains the current network state hash calculated by its sender (Section 4.1 describes the algorithm).

7.2.3. Node State TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type: NODE-STATE (5)   |   Length: > 8   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Node Identifier
|   (length fixed in DNCP profile)
|
...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Sequence Number
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Milliseconds Since Origination
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               H(Node Data)
|   (length fixed in DNCP profile)
|
...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   (optionally) Node Data (a set of nested TLVs)
|
...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

This TLV represents the local node's knowledge about the published state of a node in the DNCP network identified by the Node Identifier field in the TLV.

Every node, including the node publishing the node data, MUST update the Milliseconds Since Origination whenever it sends a Node State TLV based on when the node estimates the data was originally published. This is, e.g., to ensure that any relative timestamps contained within the published node data can be correctly offset and interpreted. Ultimately, what is provided is just an approximation, as transmission delays are not accounted for.

Absent any changes, if the originating node notices that the 32-bit milliseconds since origination value would be close to overflow (greater than $2^{32}-2^{16}$), the node MUST re-publish its TLVs even if there is no change. In other words, absent any other changes, the TLV set MUST be re-published roughly every 48 days.

The actual node data of the node may be included within the TLV as well in the optional Node Data field. The set of TLVs MUST be strictly ordered based on ascending binary content (including TLV type and length). This enables, e.g., efficient state delta processing and no-copy indexing by TLV type by the recipient. The Node Data content MUST be passed along exactly as it was received. It SHOULD be also verified on receipt that the locally calculated $H(\text{Node Data})$ matches the content of the field within the TLV, and if the hash differs, the TLV SHOULD be ignored.

7.3. Data TLVs within Node State TLV

These TLVs are published by the DNCP nodes, and therefore only encoded in the Node Data field of Node State TLVs. If encountered outside Node State TLV, they MUST be silently ignored.

7.3.1. Peer TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Type: PEER (8)          |          Length: > 8          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Peer Node Identifier          |                          |
|          (length fixed in DNCP profile)          |                          |
|          ...          |                          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Peer Endpoint Identifier          |                          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          (Local) Endpoint Identifier          |                          |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

This TLV indicates that the node in question vouches that the specified peer is reachable by it on the specified local endpoint.

The presence of this TLV at least guarantees that the node publishing it has received traffic from the peer recently. For guaranteed up-to-date bidirectional reachability, the existence of both nodes' matching Peer TLVs needs to be checked.

7.3.2. Keep-Alive Interval TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Type: KEEP-ALIVE-INTERVAL (9) | Length: >= 8 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Endpoint Identifier |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Interval |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

This TLV indicates a non-default interval being used to send keep-alives specified in Section 6.1.

Endpoint identifier is used to identify the particular (local) endpoint for which the interval applies on the sending node. If 0, it applies for ALL endpoints for which no specific TLV exists.

Interval specifies the interval in milliseconds at which the node sends keep-alives. A value of zero means no keep-alives are sent at all; in that case, some lower layer mechanism that ensures presence of nodes MUST be available and used.

8. Security and Trust Management

If specified in the DNCP profile, either DTLS [RFC6347] or TLS [RFC5246] may be used to authenticate and encrypt either some (if specified optional in the profile), or all unicast traffic. The following methods for establishing trust are defined, but it is up to the DNCP profile to specify which ones may, should or must be supported.

8.1. Pre-Shared Key Based Trust Method

A PSK-based trust model is a simple security management mechanism that allows an administrator to deploy devices to an existing network by configuring them with a pre-defined key, similar to the configuration of an administrator password or WPA-key. Although limited in nature it is useful to provide a user-friendly security mechanism for smaller networks.

8.2. PKI Based Trust Method

A PKI-based trust-model enables more advanced management capabilities at the cost of increased complexity and bootstrapping effort. It however allows trust to be managed in a centralized manner and is therefore useful for larger networks with a need for an authoritative trust management.

8.3. Certificate Based Trust Consensus Method

For some scenarios - such as bootstrapping a mostly unmanaged network - the methods described above may not provide a desirable tradeoff between security and user experience. This section includes guidance for implementing an opportunistic security [RFC7435] method which DNCP profiles can build upon and adapt for their specific requirements.

The certificate-based consensus model is designed to be a compromise between trust management effort and flexibility. It is based on X.509-certificates and allows each DNCP node to provide a trust verdict on any other certificate and a consensus is found to determine whether a node using this certificate or any certificate signed by it is to be trusted.

A DNCP node not using this security method **MUST** ignore all announced trust verdicts and **MUST NOT** announce any such verdicts by itself, i.e., any other normative language in this subsection does not apply to it.

The current effective trust verdict for any certificate is defined as the one with the highest priority from all trust verdicts announced for said certificate at the time.

8.3.1. Trust Verdicts

Trust verdicts are statements of DNCP nodes about the trustworthiness of X.509-certificates. There are 5 possible trust verdicts in order of ascending priority:

0 (Neutral): no trust verdict exists but the DNCP network should determine one.

1 (Cached Trust): the last known effective trust verdict was Configured or Cached Trust.

2 (Cached Distrust): the last known effective trust verdict was Configured or Cached Distrust.

3 (Configured Trust): trustworthy based upon an external ceremony or configuration.

4 (Configured Distrust): not trustworthy based upon an external ceremony or configuration.

Trust verdicts are differentiated in 3 groups:

- o Configured verdicts are used to announce explicit trust verdicts a node has based on any external trust bootstrap or predefined relation a node has formed with a given certificate.
- o Cached verdicts are used to retain the last known trust state in case all nodes with configured verdicts about a given certificate have been disconnected or turned off.
- o The Neutral verdict is used to announce a new node intending to join the network so a final verdict for it can be found.

The current effective trust verdict for any certificate is defined as the one with the highest priority within the set of trust verdicts announced for the certificate in the DNCP network. A node **MUST** be trusted for participating in the DNCP network if and only if the current effective trust verdict for its own certificate or any one in its certificate hierarchy is (Cached or Configured) Trust and none of the certificates in its hierarchy have an effective trust verdict of (Cached or Configured) Distrust. In case a node has a configured verdict, which is different from the current effective trust verdict for a certificate, the current effective trust verdict takes precedence in deciding trustworthiness. Despite that, the node still retains and announces its configured verdict.

8.3.2. Trust Cache

Each node **SHOULD** maintain a trust cache containing the current effective trust verdicts for all certificates currently announced in the DNCP network. This cache is used as a backup of the last known state in case there is no node announcing a configured verdict for a known certificate. It **SHOULD** be saved to a non-volatile memory at reasonable time intervals to survive a reboot or power outage.

Every time a node (re)joins the network or detects the change of an effective trust verdict for any certificate, it will synchronize its cache, i.e., store new effective trust verdicts overwriting any previously cached verdicts. Configured verdicts are stored in the cache as their respective cached counterparts. Neutral verdicts are never stored and do not override existing cached verdicts.

Verdict represents the numerical index of the trust verdict.

(reserved) is reserved for future additions and MUST be set to 0 when creating TLVs and ignored when parsing them.

SHA-256 Fingerprint contains the SHA-256 [RFC6234] hash value of the certificate in DER-format.

Common Name contains the variable-length (1-64 bytes) common name of the certificate.

8.3.4. Bootstrap Ceremonies

The following non-exhaustive list of methods describes possible ways to establish trust relationships between DNCP nodes and node certificates. Trust establishment is a two-way process in which the existing network must trust the newly added node and the newly added node must trust at least one of its peer nodes. It is therefore necessary that both the newly added node and an already trusted node perform such a ceremony to successfully introduce a node into the DNCP network. In all cases an administrator MUST be provided with external means to identify the node belonging to a certificate based on its fingerprint and a meaningful common name.

8.3.4.1. Trust by Identification

A node implementing certificate-based trust MUST provide an interface to retrieve the current set of effective trust verdicts, fingerprints and names of all certificates currently known and set configured trust verdicts to be announced. Alternatively it MAY provide a companion DNCP node or application with these capabilities with which it has a pre-established trust relationship.

8.3.4.2. Preconfigured Trust

A node MAY be preconfigured to trust a certain set of node or CA certificates. However such trust relationships MUST NOT result in unwanted or unrelated trust for nodes not intended to be run inside the same network (e.g., all other devices by the same manufacturer).

8.3.4.3. Trust on Button Press

A node MAY provide a physical or virtual interface to put one or more of its internal network interfaces temporarily into a mode in which it trusts the certificate of the first DNCP node it can successfully establish a connection with.

8.3.4.4. Trust on First Use

A node which is not associated with any other DNCP node MAY trust the certificate of the first DNCP node it can successfully establish a connection with. This method MUST NOT be used when the node has already associated with any other DNCP node.

9. DNCP Profile-Specific Definitions

Each DNCP profile MUST specify the following aspects:

- o Unicast and optionally multicast transport protocol(s) to be used. If multicast-based node and status discovery is desired, a datagram-based transport supporting multicast has to be available.
- o How the chosen transport(s) are secured: Not at all, optionally or always with the TLS scheme defined here using one or more of the methods, or with something else. If the links with DNCP nodes can be sufficiently secured or isolated, it is possible to run DNCP in a secure manner without using any form of authentication or encryption.
- o Transport protocols' parameters such as port numbers to be used, or multicast address to be used. Unicast, multicast, and secure unicast may each require different parameters, if applicable.
- o When receiving TLVs, what sort of TLVs are ignored in addition - as specified in Section 4.4 - e.g., for security reasons. While the security of the node data published within the Node State TLVs is already ensured by the base specification (if secure mode is enabled, Node State TLVs are sent only via unicast as multicast ones are ignored on receipt), if a profile adds TLVs that are sent outside the node data, a profile should indicate whether or not those TLVs should be ignored if they are received via multicast or non-secured unicast. A DNCP profile may define the following DNCP TLVs to be safely ignored:
 - * Anything received over multicast, except Node Endpoint TLV (Section 7.2.1) and Network State TLV (Section 7.2.2).
 - * Any TLVs received over unreliable unicast or multicast at too high rate; Trickle will ensure eventual convergence given the rate slows down at some point.
- o How to deal with node identifier collision as described in Section 4.4. Main options are either for one or both nodes to assign new node identifiers to themselves, or to notify someone about a fatal error condition in the DNCP network.

- o Imin, Imax and k ranges to be suggested for implementations to be used in the Trickle algorithm. The Trickle algorithm does not require these to be the same across all implementations for it to work, but similar orders of magnitude helps implementations of a DNCP profile to behave more consistently and to facilitate estimation of lower and upper bounds for convergence behavior of the network.
- o Hash function $H(x)$ to be used, and how many bits of the output are actually used. The chosen hash function is used to handle both hashing of node data, and to produce network state hash, which is a hash of node data hashes. SHA-256 defined in [RFC6234] is the recommended default choice, but a non-cryptographic hash function could be used as well. If there is a hash collision in the network state hash, the network will effectively be partitioned to partitions that believe that they are up to date, but actually no longer converged. The network will converge either when some node data anywhere in the network changes, or when conflicting Node State TLVs get transmitted across the partition (either caused by Trickle-Driven Status Updates (Section 4.3) or as part of the Processing of Received TLVs (Section 4.4)). If a node publishes node data with a hash that collides with any previously published node data, the update may not be (fully) propagated and the old version of node data may be used instead.
- o DNCP_NODE_IDENTIFIER_LENGTH: The fixed length of a node identifier (in bytes).
- o Whether to send keep-alives, and if so, whether per-endpoint (requires multicast transport), or per-peer. Keep-alive has also associated parameters:
 - * DNCP_KEEPLIVE_INTERVAL: How often keep-alives are to be sent by default (if enabled).
 - * DNCP_KEEPLIVE_MULTIPLIER: How many times the DNCP_KEEPLIVE_INTERVAL (or peer-supplied keep-alive interval value) a node may not be heard from to be considered still valid. This is just a default used in absence of any other configuration information, or particular per-endpoint configuration.
- o Whether to support dense multicast-enabled link optimization (Section 6.2) or not.

For some guidance on choosing transport and security options, please see Appendix B.

10. Security Considerations

DNCP-based protocols may use multicast to indicate DNCP state changes and for keep-alive purposes. However, no actual published data TLVs will be sent across that channel. Therefore an attacker may only learn hash values of the state within DNCP and may be able to trigger unicast synchronization attempts between nodes on a local link this way. A DNCP node MUST therefore rate-limit its reactions to multicast packets.

When using DNCP to bootstrap a network, PKI based solutions may have issues when validating certificates due to potentially unavailable accurate time, or due to inability to use the network to either check Certificate Revocation Lists or perform on-line validation.

The Certificate-based trust consensus mechanism defined in this document allows for a consenting revocation, however in case of a compromised device the trust cache may be poisoned before the actual revocation happens allowing the distrusted device to rejoin the network using a different identity. Stopping such an attack might require physical intervention and flushing of the trust caches.

11. IANA Considerations

IANA should set up a registry for the (decimal 16-bit) "DNCP TLV Types" under "Distributed Node Consensus Protocol (DNCP)", with the following initial contents: ([RFC Editor: please remove] ideally as <http://www.iana.org/assignments/dncp-registry>)

- 0: Reserved
- 1: Request network state
- 2: Request node state
- 3: Node endpoint
- 4: Network state
- 5: Node state
- 6: Reserved (was: Custom)
- 7: Reserved (was: Fragment count)
- 8: Peer
- 9: Keep-alive interval

10: Trust-Verdict

11-31: Free - policy of standards action [RFC5226] should be used

32-511: Reserved for per-DNCP profile use

512-767: Free - policy of standards action [RFC5226] should be used

768-1023: Private use [RFC5226]

1024-65535: Reserved for future protocol evolution (for example, DNCP version 2)

12. References

12.1. Normative references

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206, March 2011, <<http://www.rfc-editor.org/info/rfc6206>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<http://www.rfc-editor.org/info/rfc6234>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.

12.2. Informative references

- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <<http://www.rfc-editor.org/info/rfc3493>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<http://www.rfc-editor.org/info/rfc3315>>.

- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<http://www.rfc-editor.org/info/rfc6347>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <<http://www.rfc-editor.org/info/rfc7435>>.
- [I-D.ietf-homenet-prefix-assignment]
Pfister, P., Paterson, B., and J. Arkko, "Distributed Prefix Assignment Algorithm", draft-ietf-homenet-prefix-assignment-08 (work in progress), August 2015.

Appendix A. Alternative Modes of Operation

Beyond what is described in the main text, the protocol allows for other uses. These are provided as examples.

A.1. Read-only Operation

If a node uses just a single endpoint and does not need to publish any TLVs, full DNCP node functionality is not required. Such limited node can acquire and maintain view of the TLV space by implementing the processing logic as specified in Section 4.4. Such node would not need Trickle, peer-maintenance or even keep-alives at all, as the DNCP nodes' use of it would guarantee eventual receipt of network state hashes, and synchronization of node data, even in presence of unreliable transport.

A.2. Forwarding Operation

If a node with a pair of endpoints does not need to publish any TLVs, it can detect (for example) nodes with the highest node identifier on each of the endpoints (if any). Any TLVs received from one of them would be forwarded verbatim as unicast to the other node with highest node identifier.

Any tinkering with the TLVs would remove guarantees of this scheme working; however passive monitoring would obviously be fine. This type of simple forwarding cannot be chained, as it does not send anything proactively.

Appendix B. DNCP Profile Additional Guidance

This appendix explains implications of design choices made when specifying DNCP profile to use particular transport or security options.

B.1. Unicast Transport - UDP or TCP?

The node data published by a DNCP node is limited to 64KB due to the 16-bit size of the length field of the TLV it is published within. Some transport choices may decrease this limit; if using e.g. UDP datagrams for unicast transport the upper bound of node data size is whatever the nodes and the underlying network can pass to each other as DNCP does not define its own fragmentation scheme. A profile which chooses UDP has to be limited to small node data (e.g. somewhat smaller than IPv6 default MTU if using IPv6), or specify a minimum which all nodes have to support. Even then, if using non-link-local communications, there is some concern about what middleboxes do to fragmented packets. Therefore, the use of stream transport such as TCP is probably a good idea if either non-link-local communication is desired, or fragmentation is expected to cause problems.

TCP also provides some other facilities, such as a relatively long built-in keep-alive which in conjunction with connection closes occurring from eventual failed retransmissions may be sufficient to avoid the use of in-protocol keep-alive defined in Section 6.1. Additionally it is reliable, so there is no need for Trickle on such unicast connections.

The major downside of using TCP instead of UDP with DNCP-based profiles lies in the loss of control over the time at which TLVs are received; while unreliable UDP datagrams also have some delay, TLVs within reliable stream transport may be delayed significantly due to retransmissions. This is not a problem if no relative time dependent information is stored within the TLVs in the DNCP-based protocol; for such a protocol, TCP is a reasonable choice for unicast transport if it is available.

B.2. (Optional) Multicast Transport

Multicast is needed for dynamic peer discovery and to trigger unicast exchanges; for that, unreliable datagram transport (=typically UDP) is the only transport option defined within this specification, although DNCP-based protocols may themselves define some other transport or peer discovery mechanism (e.g. based on mDNS or DNS).

If multicast is used, a well-known address should be specified, and for e.g. IPv6 respectively the desired address scopes. In most cases link-local and possibly site-local are useful scopes.

B.3. (Optional) Transport Security

In terms of provided security, DTLS and TLS are equivalent; they also consume similar amount of state on the devices. While TLS is on top of a stream protocol, using DTLS also requires relatively long session caching within the DTLS layer to avoid expensive re-authentication/authorization steps if and when any state within the DNCP network changes or per-peer keep-alive (if enabled) is sent.

TLS implementations (at the time of the writing of the specification) seem more mature and available (as open source) than DTLS ones. This may be due to a long history of use with HTTPS.

Some libraries seem not to support multiplexing between insecure and secure communication on the same port, so specifying distinct ports for secured and unsecured communication may be beneficial.

Appendix C. Example Profile

This is the DNCP profile of SHSP, an experimental (and for the purposes of this document fictional) home automation protocol. The protocol itself is used to make key-value store published by each of the nodes available to all other nodes for distributed monitoring and control of a home infrastructure. It defines only one additional TLV type: a key=value TLV which contains a single key=value assignment for publication.

- o Unicast transport: IPv6 TCP on port EXAMPLE-P1 since only absolute timestamps are used within the key=value data and since it focuses primarily on Linux-based nodes which support both protocols well. Connections from and to non-link-local addresses are ignored to avoid exposing this protocol outside the secure links.
- o Multicast transport: IPv6 UDP on port EXAMPLE-P2 to link-local scoped multicast address ff02:EXAMPLE. At least one node per link in the home is assumed to facilitate node discovery without depending on any other infrastructure.
- o Security: None. It is to be used only on trusted links (WPA2-x wireless, physically secure wired links).
- o Additional TLVs to be ignored: None. No DNCP security is specified, and no new TLVs are defined outside of node data.

- o Node identifier length (DNCP_NODE_IDENTIFIER_LENGTH): 32 bits that are randomly generated.
- o Node identifier collision handling: Pick new random node identifier.
- o Trickle parameters: $I_{min} = 200ms$, $I_{max} = 7$, $k = 1$. It means at least one multicast per link in 25 seconds in stable state ($0.2 * 2^7$).
- o Hash function $H(x)$ + length: SHA-256, only 128 bits used. Relatively fast, and 128 bits should be plenty to prevent random conflicts (64 bits would most likely be sufficient, too).
- o No in-protocol keep-alives (Section 6.1); TCP keep-alive is to be used. In practice TCP keep-alive is seldom encountered anyway as changes in network state cause packets to be sent on the unicast connections, and those that fail sufficiently many retransmissions are dropped much before keep-alive actually would fire.
- o No support for dense multicast-enabled link optimization (Section 6.2); SHSP is a simple protocol for few nodes (network-wide, not even to mention on a single link), and therefore would not provide any benefit.

Appendix D. Some Questions and Answers [RFC Editor: please remove]

Q: 32-bit endpoint id?

A: Here, it would save 32 bits per peer if it was 16 bits (and less is not realistic). However, TLVs defined elsewhere would not seem to even gain that much on average. 32 bits is also used for ifindex in various operating systems, making for simpler implementation.

Q: Why have topology information at all?

A: It is an alternative to the more traditional seq#/TTL-based flooding schemes. In steady state, there is no need to, e.g., re-publish every now and then.

Appendix E. Changelog [RFC Editor: please remove]

draft-ietf-homenet-dncp-10:

- o Added profile guidance section, as well as example profile.

draft-ietf-homenet-dncp-09:

- o Reserved 1024+ TLV types for future versions (=versioning mechanism); private use section moved from 192-255 to 512-767.
- o Added applicability statement and clarified some text based on reviews.

draft-ietf-homenet-dncp-08:

- o Removed fragmentation as it is somewhat underspecified and unimplemented. It may be specified in some future extension draft or new version of DNCP.
- o Added generic sub-TLV extensibility mechanism.

draft-ietf-homenet-dncp-06:

- o Removed custom TLV.
- o Made keep-alive multipliers local implementation choice, profiles just provide guidance on sane default value.
- o Removed the DNCP_GRACE_INTERVAL as it is really implementation choice.
- o Simplified the suggested structures in data model.
- o Reorganized the document and provided an overview section.

draft-ietf-homenet-dncp-04:

- o Added mandatory rate limiting for network state requests, and optional slightly faster convergence mechanism by including current local network state in the remote network state requests.

draft-ietf-homenet-dncp-03:

- o Renamed connection -> endpoint.
- o !!! Backwards incompatible change: Renumbered TLVs, and got rid of node data TLV; instead, node data TLV's contents are optionally within node state TLV.

draft-ietf-homenet-dncp-02:

- o Changed DNCP "messages" into series of TLV streams, allowing optimized round-trip saving synchronization.

- o Added fragmentation support for bigger node data and for chunking in absence of reliable L2 and L3 fragmentation.

draft-ietf-homenet-dncp-01:

- o Fixed keep-alive semantics to consider unicast requests also updates of most recently consistent, and added proactive unicast request to ensure even inconsistent keep-alive messages eventually triggering consistency timestamp update.
- o Facilitated (simple) read-only clients by making Node Connection TLV optional if just using DNCP for read-only purposes.
- o Added text describing how to deal with "dense" networks, but left actual numbers and mechanics up to DNCP profiles and (local) configurations.

draft-ietf-homenet-dncp-00: Split from pre-version of draft-ietf-homenet-hncp-03 generic parts. Changes that affect implementations:

- o TLVs were renumbered.
- o TLV length does not include header (=-4). This facilitates, e.g., use of DHCPv6 option parsing libraries (same encoding), and reduces complexity (no need to handle error values of length less than 4).
- o Trickle is reset only when locally calculated network state hash is changes, not as remote different network state hash is seen. This prevents, e.g., attacks by multicast with one multicast packet to force Trickle reset on every interface of every node on a link.
- o Instead of 'ping', use 'keep-alive' (optional) for dead peer detection. Different message used!

Appendix F. Draft Source [RFC Editor: please remove]

As usual, this draft is available at <https://github.com/fingon/ietf-drafts/> in source format (with nice Makefile too). Feel free to send comments and/or pull requests if and when you have changes to it!

Appendix G. Acknowledgements

Thanks to Ole Troan, Pierre Pfister, Mark Baugher, Mark Townsley, Juliusz Chroboczek, Jiazi Yi, Mikael Abrahamsson, Brian Carpenter, Thomas Clausen, DENG Hui and Margaret Cullen for their contributions to the draft.

Thanks to Kaiwen Jin and Xavier Bonnetain for their related research work.

Authors' Addresses

Markus Stenberg
Independent
Helsinki 00930
Finland

Email: markus.stenberg@iki.fi

Steven Barth
Independent
Halle 06114
Germany

Email: cyrus@openwrt.org

HOMENET
Internet-Draft
Intended status: Standards Track
Expires: March 26, 2016

D. Migault (Ed)
Ericsson
R. Weber
Nominum
R. Hunter
Globis Consulting BV
C. Griffiths

W. Cloetens
SoftAtHome
September 23, 2015

Outsourcing Home Network Authoritative Naming Service
draft-ietf-homenet-front-end-naming-delegation-04.txt

Abstract

RFC7368 'IPv6 Home Networking Architecture Principles' section 3.7 describes architecture principles related to naming and service discovery in residential home networks.

Customer Edge Routers and other Customer Premises Equipment (CPEs) are designed to provide IP connectivity to home networks. Most CPEs assign IP addresses to the nodes of the home network which makes them good candidates for hosting the naming service. IPv6 provides global connectivity, and nodes from the home network will be reachable from the global Internet. As a result, the naming service is expected to be exposed on the Internet.

However, CPEs have not been designed to host such a naming service exposed on the Internet. Running a naming service visible on the Internet may expose the CPEs to resource exhaustion and other attacks, which could make the home network unreachable, and most probably would also affect the internal communications of the home network.

In addition, regular end users may not understand, or possess the necessary skills to be able to perform, DNSSEC management and configuration. Misconfiguration may also result in naming service disruption, thus these end users may prefer to rely on third party name service providers.

This document describes a homenet naming architecture, where the CPEs manage the DNS zones associated with its own home network, and outsource elements of the naming service (possibly including DNSSEC management) to a third party running on the Internet.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 26, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements notation	3
2. Introduction	3
2.1. Scope of the Document	4
3. Terminology	5
4. Architecture Description	7
4.1. Architecture Overview	7
4.2. Example: Homenet Zone	9
4.3. Example: CPE necessary parameters for outsourcing	11
5. Synchronization between CPE and the Synchronization Server	12
5.1. Synchronization with a Hidden Primary	12
5.2. Securing Synchronization	13
5.3. CPE Security Policies	15
6. DNSSEC compliant Homenet Architecture	15
6.1. Zone Signing	15

6.2. Secure Delegation	17
7. Handling Different Views	18
7.1. Misleading Reasons for Local Scope DNS Zone	18
7.2. Consequences	19
7.3. Guidance and Recommendations	19
8. Homenet Reverse Zone	20
9. Renumbering	20
9.1. Hidden Primary	21
9.2. Synchronization Server	22
10. Privacy Considerations	23
11. Security Considerations	23
11.1. Names are less secure than IP addresses	23
11.2. Names are less volatile than IP addresses	24
11.3. DNS Reflection Attacks	24
11.3.1. Reflection Attack involving the Hidden Primary	24
11.3.2. Reflection Attacks involving the Synchronization Server	26
11.3.3. Reflection Attacks involving the Public Authoritative Servers	27
11.4. Flooding Attack	27
11.5. Replay Attack	27
12. IANA Considerations	28
13. Acknowledgment	28
14. References	28
14.1. Normative References	28
14.2. Informational References	31
Appendix A. Document Change Log	32
Authors' Addresses	34

1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Introduction

IPv6 provides global end to end IP reachability. End users prefer to use names instead of long and complex IPv6 addresses when accessing services hosted in the home network.

Customer Edge Routers and other Customer Premises Equipment (CPEs) are already providing IPv6 connectivity to the home network, and generally provide IPv6 addresses or prefixes to the nodes of the home network. This makes CPEs good candidates to manage the binding between names and IP addresses of nodes. In addition, [RFC7368] recommends that home networks be resilient to connectivity disruption from the ISP. This could be achieved by a dedicated device inside

the home network that builds the Homenet Zone, thus providing bindings between names and IP addresses. All this makes the CPE the natural candidate for populating the Homenet zone.

CPEs are usually low powered devices designed for the home network, but not for terminating heavy traffic. As a result, hosting an authoritative DNS service on the Internet may expose the home network to resource exhaustion and other attacks. This may isolate the home network from the Internet and also impact the services hosted by the CPEs, thus affecting overall home network communication.

In order to avoid resource exhaustion and other attacks, this document describes an architecture that outsources the authoritative naming service of the home network. More specifically, the Homenet Zone built by the CPE is outsourced to an Outsourcing Infrastructure. The Outsourcing Infrastructure publishes the corresponding Public Homenet Zone on the Internet. Section 4.1 describes the architecture. In order to keep the Public Homenet Zone up-to-date Section 5 describes how the Homenet Zone and the Public Homenet Zone can be synchronized. The proposed architecture aims at deploying DNSSEC, and the Public Homenet Zone is expected to be signed with a secure delegation. The zone signing and secure delegation may be performed either by the CPE or by the Outsourcing Infrastructure. Section 6 discusses these two alternatives. Section 7 discusses the consequences of publishing multiple representations of the same zone also commonly designated as views. This section provides guidance to limit the risks associated with multiple views. Section 8 discusses management of the reverse zone. Section 9 discusses how renumbering should be handled. Finally, Section 10 and Section 11 respectively discuss privacy and security considerations when outsourcing the Homenet Zone.

2.1. Scope of the Document

The scope of the document is to describe an architecture that outsources the authoritative naming service of the home network and more specifically the interactions between the home network and the Outsourcing Infrastructure. Considerations or descriptions on inner communications or organization of the home network are provided for completeness and for clarifying the interface between the home network and the Outsourcing Infrastructure

For sake of simplicity, this document designates by "CPE" that connects the home network with the Outsourcing Infrastructure - and so performs most of the operations for outsourcing the home network naming architecture. On the other hand, CPE are usually associated to home router. These two functions - e.g routing and naming - are not correlated and could be split in multiple devices. More

specifically, this document does not consider the home network has a single CPE nor a single ISP.

In fact this document considers the CPE as a set of functionalities that can be collocated on a single device or split between multiple devices. The split of these functions between different devices as well as their potential associated communications is considered as implementation dependent and out of scope of the architecture. The only limitation this architecture considers is that the Hidden Primary be located in a single place as long as the distributed Primary architecture has not been defined. As a result, if there are multiple candidates for hosting the Hidden Primary, the home network should select a single device.

3. Terminology

- Customer Premises Equipment: (CPE) is a router providing connectivity to the home network. It might be configured and managed by the end user. In this document, the CPE might also host services such as DHCPv6. This device might be provided by the ISP. A home network may have multiple CPE, and each of them may be connected to an Internet Service Provider. In this document, the CPE represents the set of functions involved in the naming architecture. These functions may be collocated on a single device, or distributed between multiple devices. How these functions communicate is out of the scope of this document and is left for implementation. See Section 2.1 for more details.
- Registered Homenet Domain: is the Domain Name associated to the home network.
- Homenet Zone: is the DNS zone associated with the home network. It is designated by its Registered Homenet Domain. This zone is built by the CPE and contains the bindings between names and IP addresses of the nodes in the home network. The CPE synchronizes the Homenet Zone with the Synchronization Server via a hidden primary / secondary architecture. The Outsourcing Infrastructure may process the Homenet Zone - for example providing DNSSEC signing - to generate the Public Homenet Zone. This Public Homenet Zone is then transmitted to the Public Authoritative Server(s) that publish it on the Internet.
- Public Homenet Zone: is the public version of the Homenet Zone. It is expected to be signed with DNSSEC. It is hosted by the Public Authoritative Server(s), which are authoritative for this zone. The Public Homenet Zone and the Homenet Zone might be different. For example some names might not become

reachable from the Internet, and thus not be hosted in the Public Homenet Zone. Another example of difference may also occur when the Public Homenet Zone is signed whereas the Homenet Zone is not signed.

- Outsourcing Infrastructure: is the combination of the Synchronization Server and the Public Authoritative Server(s).
- Public Authoritative Servers: are the authoritative name servers hosting the Public Homenet Zone. Name resolution requests for the Homenet Domain are sent to these servers. For resiliency the Public Homenet Zone SHOULD be hosted on multiple servers.
- Synchronization Server: is the server with which the CPE synchronizes the Homenet Zone. The Synchronization Server is configured as a secondary and the CPE acts as primary. There MAY be multiple Synchronization Servers, but the text assumes a single server. In addition, the text assumes the Synchronization Server is a separate entity. This is not a requirement, and when the CPE signs the zone, the synchronization function might also be operated by the Public Authoritative Servers.
- Homenet Reverse Zone: The reverse zone file associated with the Homenet Zone.
- Reverse Public Authoritative Servers: are the authoritative name server(s) hosting the Public Homenet Reverse Zone. Queries for reverse resolution of the Homenet Domain are sent to this server. Similarly to Public Authoritative Servers, for resiliency, the Homenet Reverse Zone SHOULD be hosted on multiple servers.
- Reverse Synchronization Server: is the server with which the CPE synchronizes the Homenet Reverse Zone. It is configured as a secondary and the CPE acts as primary. There MAY be multiple Reverse Synchronization Servers, but the text assumes a single server. In addition, the text assumes the Reverse Synchronization Server is a separate entity. This is not a requirement, and when the CPE signs the zone, the synchronization function might also be operated by the Reverse Public Authoritative Servers.
- Hidden Primary: designates the primary server of the CPE, that synchronizes the Homenet Zone with the Synchronization Server. A primary / secondary architecture is used between the CPE and the Synchronization Server. The hidden primary is not expected to serve end user queries for the Homenet Zone as a regular

primary server would. The hidden primary is only known to its associated Synchronization Server.

4. Architecture Description

This section describes the architecture for outsourcing the authoritative naming service from the CPE to the Outsourcing Infrastructure. Section 4.1 describes the architecture, Section 4.2 and Section 4.3 illustrates this architecture and shows how the Homenet Zone should be built by the CPE. It also lists the necessary parameters the CPE needs to be able to outsource the authoritative naming service. These two sections are informational and non-normative.

4.1. Architecture Overview

Figure 1 provides an overview of the architecture.

The home network is designated by the Registered Homenet Domain Name -- example.com in Figure 1. The CPE builds the Homenet Zone associated with the home network. How the Homenet Zone is built is out of the scope of this document. The CPE may host or interact with multiple services to determine name-to-address mappings, such as a web GUI, DHCP [RFC6644] or mDNS [RFC6762]. These services may coexist and may be used to populate the Homenet Zone. This document assumes the Homenet Zone has been populated with domain names that are intended to be publicly published and that are publicly reachable. More specifically, names associated with services or devices that are not expected to be reachable from outside the home network or names bound to non-globally reachable IP addresses MUST NOT be part of the Homenet Zone.

Once the Homenet Zone has been built, the CPE does not host an authoritative naming service, but instead outsources it to the Outsourcing Infrastructure. The Outsourcing Infrastructure takes the Homenet Zone as an input and publishes the Public Homenet Zone. If the CPE does not sign the Homenet Zone, the Outsourcing Infrastructure may instead sign it on behalf of the CPE. Figure 1 provides a more detailed description of the Outsourcing Infrastructure, but overall, it is expected that the CPE provides the Homenet Zone. Then the Public Homenet Zone is derived from the Homenet Zone and published on the Internet.

As a result, DNS queries from the DNS resolvers on the Internet are answered by the Outsourcing Infrastructure and do not reach the CPE. Figure 1 illustrates the case of the resolution of node1.example.com.

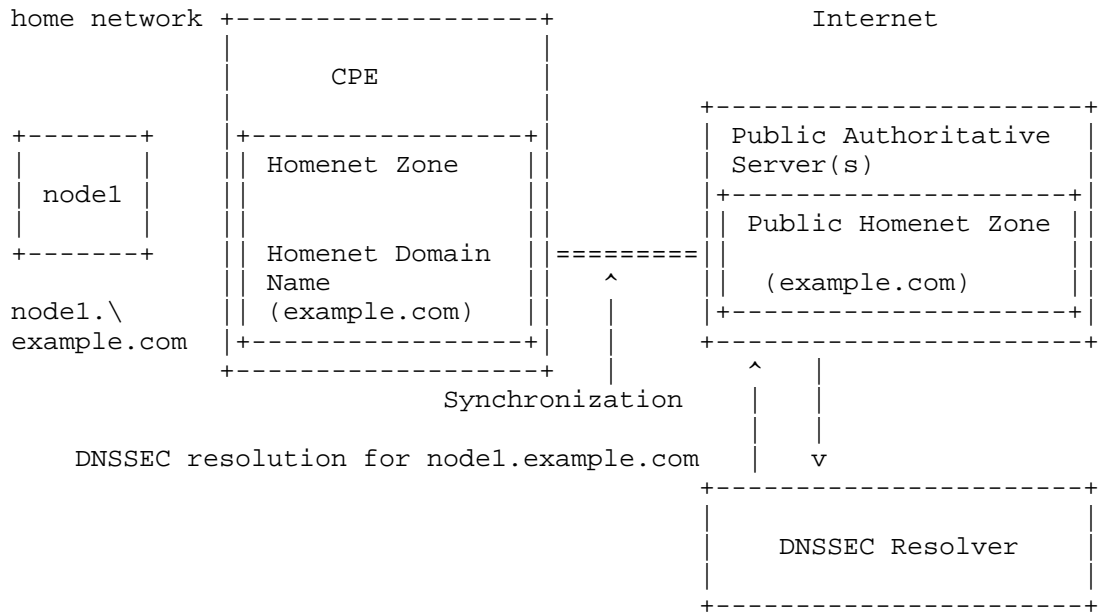


Figure 1: Homenet Naming Architecture Description

The Outsourcing Infrastructure is described in Figure 2. The Synchronization Server receives the Homenet Zone as an input. The received zone may be transformed to output the Public Homenet Zone. Various operations may be performed here, however this document only considers zone signing as a potential operation. This should occur only when the CPE outsources this operation to the Synchronization Server. On the other hand, if the CPE signs the Homenet Zone itself, the zone would be collected by the Synchronization Server and directly transferred to the Public Authoritative Server(s). These policies are discussed and detailed in Section 6 and Section 7.

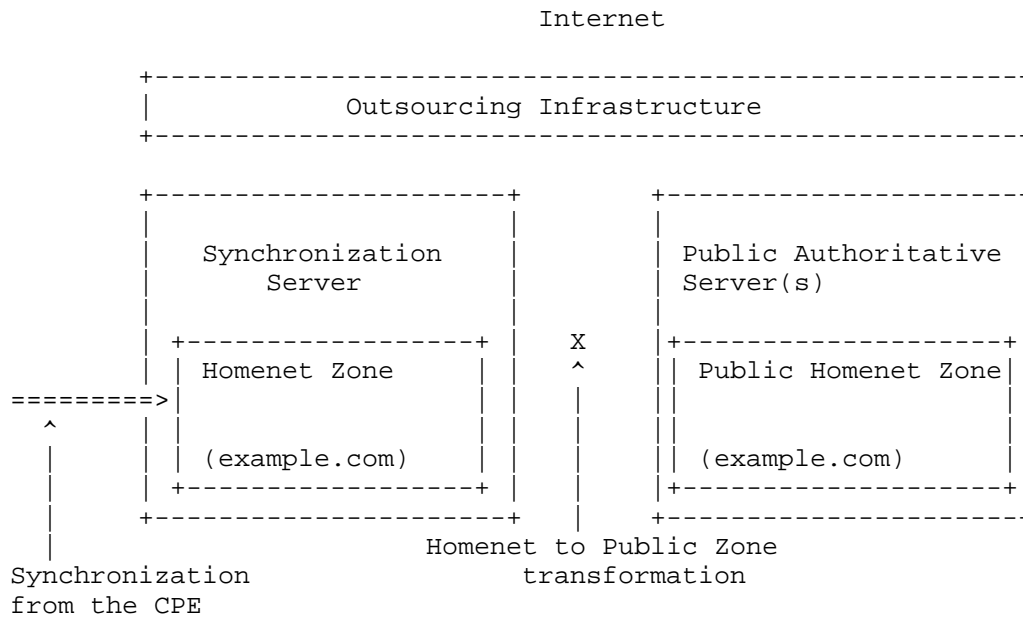


Figure 2: Outsourcing Infrastructure Description

4.2. Example: Homenet Zone

This section is not normative and intends to illustrate how the CPE builds the Homenet Zone.

As depicted in Figure 1 and Figure 2, the Public Homenet Zone is hosted on the Public Authoritative Server(s), whereas the Homenet Zone is hosted on the CPE. Motivations for keeping these two zones identical are detailed in Section 7, and this section considers that the CPE builds the zone that will be effectively published on the Public Authoritative Server(s). In other words "Homenet to Public Zone transformation" is the identity also commonly designated as "no operation" (NOP).

In that case, the Homenet Zone should configure its Name Server RRset (NS) and Start of Authority (SOA) with the values associated with the Public Authoritative Server(s). This is illustrated in Figure 3. `public.primary.example.net` is the FQDN of the Public Authoritative Server(s), and `IP1`, `IP2`, `IP3`, `IP4` are the associated IP addresses. Then the CPE should add the additional new nodes that enter the home network, remove those that should be removed, and sign the Homenet Zone.

```
$ORIGIN example.com
$TTL 1h

@ IN SOA public.primary.example.net
    hostmaster.example.com. (
        2013120710 ; serial number of this zone file
        1d         ; secondary refresh
        2h         ; secondary retry time in case of a problem
        4w         ; secondary expiration time
        1h         ; maximum caching time in case of failed
                   ; lookups
    )

@ NS public.authoritative.servers.example.net

public.primary.example.net A @IP1
public.primary.example.net A @IP2
public.primary.example.net AAAA @IP3
public.primary.example.net AAAA @IP4
```

Figure 3: Homenet Zone

The SOA RRset is defined in [RFC1033], [RFC1035] and [RFC2308]. This SOA is specific, as it is used for the synchronization between the Hidden Primary and the Synchronization Server and published on the DNS Public Authoritative Server(s)..

- MNAME: indicates the primary. In our case the zone is published on the Public Authoritative Server(s), and its name MUST be included. If multiple Public Authoritative Server(s) are involved, one of them MUST be chosen. More specifically, the CPE MUST NOT include the name of the Hidden Primary.
- RNAME: indicates the email address to reach the administrator. [RFC2142] recommends using hostmaster@domain and replacing the '@' sign by '.'.
- REFRESH and RETRY: indicate respectively in seconds how often secondaries need to check the primary, and the time between two refresh when a refresh has failed. Default values indicated by [RFC1033] are 3600 (1 hour) for refresh and 600 (10 minutes) for retry. This value might be too long for highly dynamic content. However, the Public Authoritative Server(s) and the CPE are expected to implement NOTIFY [RFC1996]. So whilst shorter refresh timers might increase the bandwidth usage for secondaries hosting large number of zones, it will have little practical impact on the elapsed time required to achieve

synchronization between the Outsourcing Infrastructure and the Hidden Master. As a result, the default values are acceptable.

EXPIRE: is the upper limit data SHOULD be kept in absence of refresh. The default value indicated by [RFC1033] is 3600000 (approx. 42 days). In home network architectures, the CPE provides both the DNS synchronization and the access to the home network. This device may be plugged and unplugged by the end user without notification, thus we recommend a long expiry timer.

MINIMUM: indicates the minimum TTL. The default value indicated by [RFC1033] is 86400 (1 day). For home network, this value MAY be reduced, and 3600 (1 hour) seems more appropriate.

4.3. Example: CPE necessary parameters for outsourcing

This section specifies the various parameters required by the CPE to configure the naming architecture of this document. This section is informational, and is intended to clarify the information handled by the CPE and the various settings to be done.

Synchronization Server may be configured with the following parameters. These parameters are necessary to establish a secure channel between the CPE and the Synchronization Server as well as to specify the DNS zone that is in the scope of the communication:

- Synchronization Server: The associated FQDNs or IP addresses of the Synchronization Server. IP addresses are optional and the FQDN is sufficient. To secure the binding name and IP addresses, a DNSSEC exchange is required. Otherwise, the IP addresses should be entered manually.
- Authentication Method: How the CPE authenticates itself to the Synchronization Server. This MAY depend on the implementation but this should cover at least IPsec, DTLS and TSIG
- Authentication data: Associated Data. PSK only requires a single argument. If other authentication mechanisms based on certificates are used, then CPE private keys, certificates and certification authority should be specified.
- Public Authoritative Server(s): The FQDN or IP addresses of the Public Authoritative Server(s). It MAY correspond to the data that will be set in the NS RRsets and SOA of the Homenet Zone. IP addresses are optional and the FQDN is sufficient. To secure the binding between name and IP addresses, a DNSSEC

exchange is required. Otherwise, the IP addresses should be entered manually.

- Registered Homenet Domain: The domain name used to establish the secure channel. This name is used by the Synchronization Server and the CPE for the primary / secondary configuration as well as to index the NOTIFY queries of the CPE when the CPE has been renumbered.

Setting the Homenet Zone requires the following information.

- Registered Homenet Domain: The Domain Name of the zone. Multiple Registered Homenet Domains may be provided. This will generate the creation of multiple Public Homenet Zones.
- Public Authoritative Server(s): The Public Authoritative Server(s) associated with the Registered Homenet Domain. Multiple Public Authoritative Server(s) may be provided.

5. Synchronization between CPE and the Synchronization Server

The Homenet Reverse Zone and the Homenet Zone MAY be updated either with DNS UPDATE [RFC2136] or using a primary / secondary synchronization. The primary / secondary mechanism is preferred as it scales better and avoids DoS attacks: First the primary notifies the secondary that the zone must be updated and leaves the secondary to proceed with the update when possible. Then, a NOTIFY message is sent by the primary, which is a small packet that is less likely to load the secondary. Finally, the AXFR query performed by the secondary is a small packet sent over TCP (section 4.2 [RFC5936]), which mitigates reflection attacks using a forged NOTIFY. On the other hand, DNS UPDATE (which can be transported over UDP), requires more processing than a NOTIFY, and does not allow the server to perform asynchronous updates.

This document RECOMMENDS use of a primary / secondary mechanism instead of the use of DNS UPDATE. This section details the primary / secondary mechanism.

5.1. Synchronization with a Hidden Primary

Uploading and dynamically updating the zone file on the Synchronization Server can be seen as zone provisioning between the CPE (Hidden Primary) and the Synchronization Server (Secondary Server). This can be handled either in band or out of band.

Note that there is no standard way to distribute a DNS primary between multiple devices. As a result, if multiple CPEs are

candidate for hosting the Hidden Primary, some specific mechanisms should be designed so the home network only selects a single CPE for the Hidden Primary. Selection mechanisms based on HNCP are good candidates [XXX].

The Synchronization Server is configured as a secondary for the Homenet Domain Name. This secondary configuration has been previously agreed between the end user and the provider of the Synchronization Server. In order to set the primary / secondary architecture, the CPE acts as a Hidden Primary Server, which is a regular authoritative DNS Server listening on the WAN interface.

The Hidden Primary Server SHOULD accept SOA [RFC1033], AXFR [RFC1034], and IXFR [RFC1995] queries from its configured secondary DNS server(s). The Hidden Primary Server SHOULD send NOTIFY messages [RFC1996] in order to update Public DNS server zones as updates occur. Because, the Homenet Zones are likely to be small, the CPE MUST implement AXFR and SHOULD implement IXFR.

Hidden Primary Server differs from a regular authoritative server for the home network by:

- Interface Binding: the Hidden Primary Server listens on the WAN Interface, whereas a regular authoritative server for the home network would listen on the home network interface.
- Limited exchanges: the purpose of the Hidden Primary Server is to synchronize with the Synchronization Server, not to serve any zones to end users. As a result, exchanges are performed with specific nodes (the Synchronization Server). Further, exchange types are limited. The only legitimate exchanges are: NOTIFY initiated by the Hidden Primary and IXFR or AXFR exchanges initiated by the Synchronization Server. On the other hand, regular authoritative servers would respond to any hosts, and any DNS query would be processed. The CPE SHOULD filter IXFR/AXFR traffic and drop traffic not initiated by the Synchronization Server. The CPE MUST listen for DNS on TCP and UDP and MUST at least allow SOA lookups of the Homenet Zone.

5.2. Securing Synchronization

Exchange between the Synchronization Server and the CPE MUST be secured, at least for integrity protection and for authentication.

TSIG [RFC2845] or SIG(0) [RFC2931] MAY be used to secure the DNS communications between the CPE and the Synchronization Server. TSIG uses a symmetric key which can be managed by TKEY [RFC2930]. Management of the key involved in SIG(0) is performed through zone

updates. How keys are rolled over with SIG(0) is out-of-scope of this document. The advantage of these mechanisms is that they are only associated with the DNS application. Not relying on shared libraries eases testing and integration. On the other hand, using TSIG, TKEY or SIG(0) requires these mechanisms to be implemented on the CPE, which adds code and complexity. Another disadvantage is that TKEY does not provide authentication mechanisms.

Protocols like TLS [RFC5246] / DTLS [RFC6347] MAY be used to secure the transactions between the Synchronization Server and the CPE. The advantage of TLS/DTLS is that this technology is widely deployed, and most of the devices already embed TLS/DTLS libraries, possibly also taking advantage of hardware acceleration. Further, TLS/DTLS provides authentication facilities and can use certificates to authenticate the Synchronization Server and the CPE. On the other hand, using TLS/DTLS requires implementing DNS exchanges over TLS/DTLS, as well as a new service port. This document therefore does NOT RECOMMEND this option.

IPsec [RFC4301] IKEv2 [RFC7296] MAY also be used to secure transactions between the CPE and the Synchronization Server. Similarly to TLS/DTLS, most CPEs already embed an IPsec stack, and IKEv2 supports multiple authentication mechanisms via the EAP framework. In addition, IPsec can be used to protect DNS exchanges between the CPE and the Synchronization Server without any modifications of the DNS server or client. DNS integration over IPsec only requires an additional security policy in the Security Policy Database (SPD). One disadvantage of IPsec is that NATs and firewall traversal may be problematic. However, in our case, the CPE is connected to the Internet, and IPsec communication between the CPE and the Synchronization Server should not be impacted by middle boxes.

How the PSK can be used by any of the TSIG, TLS/DTLS or IPsec protocols: Authentication based on certificates implies a mutual authentication and thus requires the CPE to manage a private key, a public key, or certificates, as well as Certificate Authorities. This adds complexity to the configuration especially on the CPE side. For this reason, we RECOMMEND that the CPE MAY use PSK or certificate base authentication, and that the Synchronization Server MUST support PSK and certificate based authentication.

Note also that authentication of message exchanges between the CPE and the Synchronization Server SHOULD NOT use the external IP address of the CPE to index the appropriate keys. As detailed in Section 9, the IP addresses of the Synchronization Server and the Hidden Primary are subject to change, for example while the network is being renumbered. This means that the necessary keys to authenticate

transaction SHOULD NOT be indexed using the IP address, and SHOULD be resilient to IP address changes.

5.3. CPE Security Policies

This section details security policies related to the Hidden Primary / Secondary synchronization.

The Hidden Primary, as described in this document SHOULD drop any queries from the home network. This could be implemented via port binding and/or firewall rules. The precise mechanism deployed is out of scope of this document.

The Hidden Primary SHOULD drop any DNS queries arriving on the WAN interface that are not issued from the Synchronization Server.

The Hidden Primary SHOULD drop any outgoing packets other than DNS NOTIFY query, SOA response, IXFR response or AXFR responses.

The Hidden Primary SHOULD drop any incoming packets other than DNS NOTIFY response, SOA query, IXFR query or AXFR query.

The Hidden Primary SHOULD drop any non protected IXFR or AXFR exchange, depending on how the synchronization is secured.

6. DNSSEC compliant Homenet Architecture

[RFC7368] in Section 3.7.3 recommends DNSSEC to be deployed on both the authoritative server and the resolver. The resolver side is out of scope of this document, and only the authoritative part of the server is considered.

Deploying DNSSEC requires signing the zone and configuring a secure delegation. As described in Section 4.1, signing can be performed either by the CPE or by the Outsourcing Infrastructure. Section 6.1 details the implications of these two alternatives. Similarly, the secure delegation can be performed by the CPE or by the Outsourcing Infrastructure. Section 6.2 discusses these two alternatives.

6.1. Zone Signing

This section discusses the pros and cons when zone signing is performed by the CPE or by the Outsourcing Infrastructure. It is RECOMMENDED that the CPE signs the zone unless there is a strong argument against this, such as a CPE that is not capable of signing the zone. In that case zone signing MAY be performed by the Outsourcing Infrastructure on behalf of the CPE.

Reasons for signing the zone by the CPE are:

- 1: Keeping the Homenet Zone and the Public Homenet Zone equal to securely optimize DNS resolution. As the Public Zone is signed with DNSSEC, RRsets are authenticated, and thus DNS responses can be validated even though they are not provided by the authoritative server. This provides the CPE the ability to respond on behalf of the Public Authoritative Server(s). This could be useful for example if, in the future, the CPE announces to the home network that the CPE can act as a local authoritative primary or equivalent for the Homenet Zone. Currently the CPE is not expected to receive authoritative DNS queries, as its IP address is not mentioned in the Public Homenet Zone. On the other hand most CPEs host a resolving function, and could be configured to perform a local lookup to the Homenet Zone instead of initiating a DNS exchange with the Public Authoritative Server(s). Note that outsourcing the zone signing operation means that all DNSSEC queries SHOULD be cached to perform a local lookup, otherwise a resolution with the Public Authoritative Server(s) would be performed.
- 2: Keeping the Homenet Zone and the Public Homenet Zone equal to securely address the connectivity disruption independence detailed in [RFC7368] section 4.4.1 and 3.7.5. As local lookups are possible in case of network disruption, communications within the home network can still rely on the DNSSEC service. Note that outsourcing the zone signing operation does not address connectivity disruption independence with DNSSEC. Instead local lookup would provide DNS as opposed to DNSSEC responses provided by the Public Authoritative Server(s).
- 3: Keeping the Homenet Zone and the Public Homenet Zone equal to guarantee coherence between DNS responses. Using a unique zone is one way to guarantee uniqueness of the responses among servers and places. Issues generated by different views are discussed in more details in Section 7.
- 2: Privacy and Integrity of the DNSSEC Homenet Zone are better guaranteed. When the Zone is signed by the CPE, it makes modification of the DNS data -- for example for flow redirection -- impossible. As a result, signing the Homenet Zone by the CPE provides better protection for end user privacy.

Reasons for signing the zone by the Outsourcing Infrastructure are:

- 1: The CPE may not be capable of signing the zone, most likely because its firmware does not support this function. However this reason is expected to become less and less valid over time.
- 2: Outsourcing DNSSEC management operations. Management operations involve key roll-over, which can be performed automatically by the CPE and transparently for the end user. Avoiding DNSSEC management is mostly motivated by bad software implementations.
- 3: Reducing the impact of CPE replacement on the Public Homenet Zone. Unless the CPE private keys can be extracted and stored off-device, CPE hardware replacement will result in an emergency key roll-over. This can be mitigated by using relatively small TTLs.
- 4: Reducing configuration impact on the end user. Unless there are zero configuration mechanisms in place to provide credentials between the new CPE and the Synchronization Server, authentication associations between the CPE and the Synchronization Server would need to be re-configured. As CPE replacement is not expected to happen regularly, end users may not be at ease with such configuration settings. However, mechanisms as described in [I-D.ietf-homenet-naming-architecture-dhc-options] use DHCP Options to outsource the configuration and avoid this issue.
- 5: The Outsourcing Infrastructure is more likely to handle private keys more securely than the CPE. However, having all private keys in one place may also nullify that benefit.

6.2. Secure Delegation

Secure delegation is achieved only if the DS RRset is properly set in the parent zone. Secure delegation can be performed by the CPE or the Outsourcing Infrastructures (that is the Synchronization Server or the Public Authoritative Server(s)).

The DS RRset can be updated manually with nsupdate for example. This requires the CPE or the Outsourcing Infrastructure to be authenticated by the DNS server hosting the parent of the Public Homenet Zone. Such a trust channel between the CPE and the parent DNS server may be hard to maintain with CPEs, and thus may be easier to establish with the Outsourcing Infrastructure. In fact, the Public Authoritative Server(s) may use Automating DNSSEC Delegation Trust Maintenance [RFC7344].

7. Handling Different Views

The Homenet Zone provides information about the home network. Some users may be tempted to have provide responses dependent on the origin of the DNS query. More specifically, some users may be tempted to provide a different view for DNS queries originating from the home network and for DNS queries coming from the Internet. Each view could then be associated with a dedicated Homenet Zone. Note that this document does not specify how DNS queries originating from the home network are addressed to the Homenet Zone. This could be done via hosting the DNS resolver on the CPE for example.

This section is not normative. Section 7.1 details why some nodes may only be reachable from the home network and not from the global Internet. Section 7.2 briefly describes the consequences of having distinct views such as a "home network view" and an "Internet view". Finally, Section 7.3 provides guidance on how to resolve names that are only significant in the home network, without creating different views.

7.1. Misleading Reasons for Local Scope DNS Zone

The motivation for supporting different views is to provide different answers dependent on the origin of the DNS query, for reasons such as:

- 1: An end user may want to have services not published on the Internet. Services like the CPE administration interface that provides the GUI to administer your CPE might not seem advisable to publish on the Internet. Similarly, services like the mapper that registers the devices of your home network may also not be desirable to be published on the Internet. In both cases, these services should only be known or used by the network administrator. To restrict the access of such services, the home network administrator may choose to publish these pieces of information only within the home network, where it might be assumed that the users are more trusted than on the Internet. Even though this assumption may not be valid, at least this may reduce the surface of any attack.
- 2: Services within the home network may be reachable using non global IP addresses. IPv4 and NAT may be one reason. On the other hand IPv6 may favor link-local or site-local IP addresses. These IP addresses are not significant outside the boundaries of the home network. As a result, they MAY be published in the home network view, and SHOULD NOT be published in the Public Homenet Zone.

7.2. Consequences

Enabling different views leads to a non-coherent naming system. Depending on where resolution is performed, some services will not be available. This may be especially inconvenient with devices with multiple interfaces that are attached both to the Internet via a 3G/4G interface and to the home network via a WLAN interface. Devices may also cache the results of name resolution, and these cached entries may no longer be valid if a mobile device moves between a homenet connection and an internet connection e.g. a device temporarily loses wifi signal and switches to 3G.

Regarding local-scope IP addresses, such devices may end up with poor connectivity. Suppose, for example, that DNS resolution is performed via the WLAN interface attached to the CPE, and the response provides local-scope IP addresses, but the communication is initiated on the 3G/4G interface. Communications with local-scope addresses will be unreachable on the Internet, thus aborting the communication. The same situation occurs if a device is flip / flopping between various WLAN networks.

Regarding DNSSEC, if the CPE does not sign the Homenet Zone and outsources the signing process, the two views are different, because one is protected with DNSSEC whereas the other is not. Devices with multiple interfaces will have difficulty securing the naming resolution, as responses originating from the home network may not be signed.

For devices with all its interfaces attached to a single administrative domain, that is to say the home network, or the Internet. Incoherence between DNS responses may still also occur if the device is able to perform DNS resolutions both using the DNS resolving server of the home network, or one of the ISP. DNS resolution performed via the CPE or the ISP resolver may be different than those performed over the Internet.

7.3. Guidance and Recommendations

As documented in Section 7.2, it is RECOMMENDED to avoid different views. If network administrators choose to implement multiple views, impacts on devices' resolution SHOULD be evaluated.

As a consequence, the Homenet Zone is expected to be an exact copy of the Public Homenet Zone. As a result, services that are not expected to be published on the Internet SHOULD NOT be part of the Homenet Zone, local-scope addresses SHOULD NOT be part of the Homenet Zone, and when possible, the CPE SHOULD sign the Homenet Zone.

The Homenet Zone is expected to host public information only. It is not the scope of the DNS service to define local home network boundaries. Instead, local scope information is expected to be provided to the home network using local scope naming services. mDNS [RFC6762] DNS-SD [RFC6763] are two examples of these services. Currently mDNS is limited to a single link network. However, future protocols are expected to leverage this constraint as pointed out in [RFC7558].

8. Homenet Reverse Zone

This section is focused on the Homenet Reverse Zone.

Firstly, all considerations for the Homenet Zone apply to the Homenet Reverse Zone. The main difference between the Homenet Reverse Zone and the Homenet Zone is that the parent zone of the Homenet Reverse Zone is most likely managed by the ISP. As the ISP also provides the IP prefix to the CPE, it may be able to authenticate the CPE using mechanisms outside the scope of this document e.g. the physical attachment point to the ISP network. If the Reverse Synchronization Server is managed by the ISP, credentials to authenticate the CPE for the zone synchronization may be set automatically and transparently to the end user. [I-D.ietf-homenet-naming-architecture-dhc-options] describes how automatic configuration may be performed.

With IPv6, the domain space for IP addresses is so large that reverse zone may be confronted with scalability issues. How the reverse zone is generated is out of scope of this document. [I-D.howard-dnsop-ip6rdns] provides guidance on how to address scalability issues.

9. Renumbering

This section details how renumbering is handled by the Hidden Primary server or the Synchronization Server. Both types of renumbering are discussed i.e. "make-before-break" and "break-before-make".

In the make-before-break renumbering scenario, the new prefix is advertised, the network is configured to prepare the transition to the new prefix. During a period of time, the two prefixes old and new coexist, before the old prefix is completely removed. In the break-before-make renumbering scenario, the new prefix is advertised making the old prefix obsolete.

Renumbering has been extensively described in [RFC4192] and analyzed in [RFC7010] and the reader is expected to be familiar with them before reading this section.

9.1. Hidden Primary

In a renumbering scenario, the Hidden Primary is informed it is being renumbered. In most cases, this occurs because the whole home network is being renumbered. As a result, the Homenet Zone will also be updated. Although the new and old IP addresses may be stored in the Homenet Zone, we recommend that only the newly reachable IP addresses be published.

To avoid reachability disruption, IP connectivity information provided by the DNS SHOULD be coherent with the IP plane. In our case, this means the old IP address SHOULD NOT be provided via the DNS when it is not reachable anymore. Let for example TTL be the TTL associated with a RRset of the Homenet Zone, it may be cached for TTL seconds. Let T_NEW be the time the new IP address replaces the old IP address in the Homenet Zone, and $T_OLD_UNREACHABLE$ the time the old IP is not reachable anymore. In the case of the make-before-break, seamless reachability is provided as long as $T_OLD_UNREACHABLE - T_NEW > 2 * TTL$. If this is not satisfied, then devices associated with the old IP address in the home network may become unreachable for $2 * TTL - (T_OLD_UNREACHABLE - T_NEW)$. In the case of a break-before-make, $T_OLD_UNREACHABLE = T_NEW$, and the device may become unreachable up to $2 * TTL$.

Once the Homenet Zone file has been updated on the Hidden Primary, the Hidden Primary needs to inform the Outsourcing Infrastructure that the Homenet Zone has been updated and that the IP address to use to retrieve the updated zone has also been updated. Both notifications are performed using regular DNS exchanges. Mechanisms to update an IP address provided by lower layers with protocols like SCTP [RFC4960], MOBIKE [RFC4555] are not considered in this document.

The Hidden Primary SHOULD inform the Synchronization Server that the Homenet Zone has been updated by sending a NOTIFY payload with the new IP address. In addition, this NOTIFY payload SHOULD be authenticated using SIG(0) or TSIG. When the Synchronization Server receives the NOTIFY payload, it MUST authenticate it. Note that the cryptographic key used for the authentication SHOULD be indexed by the Registered Homenet Domain contained in the NOTIFY payload as well as the RRSIG. In other words, the IP address SHOULD NOT be used as an index. If authentication succeeds, the Synchronization Server MUST also notice the IP address has been modified and perform a reachability check before updating its primary configuration. The routability check MAY be performed by sending a SOA request to the Hidden Primary using the source IP address of the NOTIFY. This exchange is also secured, and if an authenticated response is received from the Hidden Primary with the new IP address, the

Synchronization Server SHOULD update its configuration file and retrieve the Homenet Zone using an AXFR or a IXFR exchange.

Note that the primary reason for providing the IP address is that the Hidden Primary is not publicly announced in the DNS. If the Hidden Primary were publicly announced in the DNS, then the IP address update could have been performed using the DNS as described in Section 9.2.

9.2. Synchronization Server

Renumbering of the Synchronization Server results in the Synchronization Server changing its IP address. The Synchronization Server is a secondary, so its renumbering does not impact the Homenet Zone. In fact, exchanges to the Synchronization Server are restricted to the Homenet Zone synchronization. In our case, the Hidden Primary MUST be able to send NOTIFY payloads to the Synchronization Server.

If the Synchronization Server is configured in the Hidden Primary configuration file using a FQDN, then the update of the IP address is performed by DNS. More specifically, before sending the NOTIFY, the Hidden Primary performs a DNS resolution to retrieve the IP address of the secondary.

As described in Section 9.1, the Synchronization Server DNS information SHOULD be coherent with the IP plane. Let TTL be the TTL associated with the Synchronization Server FQDN, T_NEW the time the new IP address replaces the old one and T_OLD_UNREACHABLE the time the Synchronization Server is not reachable anymore with its old IP address. Seamless reachability is provided as long as $T_OLD_UNREACHABLE - T_NEW > 2 * TTL$. If this condition is not met, the Synchronization Server may be unreachable during $2 * TTL - (T_OLD_UNREACHABLE - T_NEW)$. In the case of a break-before-make, $T_OLD_UNREACHABLE = T_NEW$, and it may become unreachable up to $2 * TTL$.

Some DNS infrastructure uses the IP address to designate the secondary, in which case, other mechanisms must be found. The reason for using IP addresses instead of names is generally to reach an internal interface that is not designated by a FQDN, and to avoid potential bootstrap problems. Such scenarios are considered as out of scope in the case of home networks.

10. Privacy Considerations

Outsourcing the DNS Authoritative service from the CPE to a third party raises a few privacy related concerns.

The Homenet Zone contains a full description of the services hosted in the network. These services may not be expected to be publicly shared although their names remain accessible through the Internet. Even though DNS makes information public, the DNS does not expect to make the complete list of services public. In fact, making information public still requires the key (or FQDN) of each service to be known by the resolver in order to retrieve information about the services. More specifically, making mywebsite.example.com public in the DNS, is not sufficient to make resolvers aware of the existence web site. However, an attacker may walk the reverse DNS zone, or use other reconnaissance techniques to learn this information as described in [I-D.ietf-opsec-ipv6-host-scanning].

In order to prevent the complete Homenet Zone being published on the Internet, AXFR queries SHOULD be blocked on the Public Authoritative Server(s). Similarly, to avoid zone-walking NSEC3 [RFC5155] SHOULD be preferred over NSEC [RFC4034].

When the Homenet Zone is outsourced, the end user should be aware that it provides a complete description of the services available on the home network. More specifically, names usually provides a clear indication of the service and possibly even the device type, and as the Homenet Zone contains the IP addresses associated with the service, they also limit the scope of the scan space.

In addition to the Homenet Zone, the third party can also monitor the traffic associated with the Homenet Zone. This traffic may provide an indication of the services an end user accesses, plus how and when they use these services. Although, caching may obfuscate this information inside the home network, it is likely that outside your home network this information will not be cached.

11. Security Considerations

The Homenet Naming Architecture described in this document solves exposing the CPE's DNS service as a DoS attack vector.

11.1. Names are less secure than IP addresses

This document describes how an end user can make their services and devices from his home network reachable on the Internet by using names rather than IP addresses. This exposes the home network to attackers, since names are expected to include less entropy than IP

addresses. In fact, with IP addresses, the Interface Identifier is 64 bits long leading to up to 2^{64} possibilities for a given subnetwork. This is not to mention that the subnet prefix is also of 64 bits long, thus providing up to 2^{64} possibilities. On the other hand, names used either for the home network domain or for the devices present less entropy (livebox, router, printer, nicolas, jennifer, ...) and thus potentially exposes the devices to dictionary attacks.

11.2. Names are less volatile than IP addresses

IP addresses may be used to locate a device, a host or a service. However, home networks are not expected to be assigned a time invariant prefix by ISPs. As a result, observing IP addresses only provides some ephemeral information about who is accessing the service. On the other hand, names are not expected to be as volatile as IP addresses. As a result, logging names over time may be more valuable than logging IP addresses, especially to profile an end user's characteristics.

PTR provides a way to bind an IP address to a name. In that sense, responding to PTR DNS queries may affect the end user's privacy. For that reason end users may choose not to respond to PTR DNS queries and MAY instead return a NXDOMAIN response.

11.3. DNS Reflection Attacks

An attacker performs a reflection attack when it sends traffic to one or more intermediary nodes (reflectors), that in turn send back response traffic to the victim. Motivations for using an intermediary node might be anonymity of the attacker, as well as amplification of the traffic. Typically, when the intermediary node is a DNSSEC server, the attacker sends a DNSSEC query and the victim is likely to receive a DNSSEC response. This section analyzes how the different components may be involved as a reflector in a reflection attack. Section 11.3.1 considers the Hidden Primary, Section 11.3.2 the Synchronization Server, and Section 11.3.3 the Public Authoritative Server(s).

11.3.1. Reflection Attack involving the Hidden Primary

With the specified architecture, the Hidden Primary is only expected to receive DNS queries of type SOA, AXFR or IXFR. This section analyzes how these DNS queries may be used by an attacker to perform a reflection attack.

DNS queries of type AXFR and IXFR use TCP and as such are less subject to reflection attacks. This makes SOA queries the only

remaining practical vector of attacks for reflection attacks, based on UDP.

SOA queries are not associated with a large amplification factor compared to queries of type "ANY" or to query of non existing FQDNs. This reduces the probability a DNS query of type SOA will be involved in a DDoS attack.

SOA queries are expected to follow a very specific pattern, which makes rate limiting techniques an efficient way to limit such attacks, and associated impact on the naming service of the home network.

Motivations for such a flood might be a reflection attack, but could also be a resource exhaustion attack performed against the Hidden Primary. The Hidden Primary only expects to exchange traffic with the Synchronization Server, that is its associated secondary. Even though secondary servers may be renumbered as mentioned in Section 9, the Hidden Primary is likely to perform a DNSSEC resolution and find out the associated secondary's IP addresses in use. As a result, the Hidden Primary is likely to limit the origin of its incoming traffic based on the origin IP address.

With filtering rules based on IP address, SOA flooding attacks are limited to forged packets with the IP address of the secondary server. In other words, the only victims are the Hidden Primary itself or the secondary. There is a need for the Hidden Primary to limit that flood to limit the impact of the reflection attack on the secondary, and to limit the resource needed to carry on the traffic by the CPE hosting the Hidden Primary. On the other hand, mitigation should be performed appropriately, so as to limit the impact on the legitimate SOA sent by the secondary.

The main reason for the Synchronization Server sending a SOA query is to update the SOA RRset after the TTL expires, to check the serial number upon the receipt of a NOTIFY query from the Hidden Primary, or to re-send the SOA request when the response has not been received. When a flood of SOA queries is received by the Hidden Primary, the Hidden Primary may assume it is involved in an attack.

There are few legitimate time slots when the secondary is expected to send a SOA query. Suppose T_{NOTIFY} is the time a NOTIFY is sent by the Hidden Primary, T_{SOA} the last time the SOA has been queried, TTL the TTL associated to the SOA, and $T_{REFRESH}$ the refresh time defined in the SOA RRset. The specific time SOA queries are expected can be for example T_{NOTIFY} , $T_{SOA} + 2/3 \text{ TTL}$, $T_{SOA} + TTL$, $T_{SOA} + T_{REFRESH}$, and. Outside a few minutes following these specific time slots, the probability that the CPE discards a legitimate SOA query

is very low. Within these time slots, the probability the secondary may have its legitimate query rejected is higher. If a legitimate SOA is discarded, the secondary will re-send SOA query every "retry time" second until "expire time" seconds occurs, where "retry time" and "expire time" have been defined in the SOA.

As a result, it is RECOMMENDED to set rate limiting policies to protect CPE resources. If a flood lasts more than the expired time defined by the SOA, it is RECOMMENDED to re-initiate a synchronization between the Hidden Primary and the secondaries.

11.3.2. Reflection Attacks involving the Synchronization Server

The Synchronization Server acts as a secondary coupled with the Hidden Primary. The secondary expects to receive NOTIFY query, SOA responses, AXFR and IXFR responses from the Hidden Primary.

Sending a NOTIFY query to the secondary generates a NOTIFY response as well as initiating an SOA query exchange from the secondary to the Hidden Primary. As mentioned in [RFC1996], this is a known "benign denial of service attack". As a result, the Synchronization Server SHOULD enforce rate limiting on sending SOA queries and NOTIFY responses to the Hidden Primary. Most likely, when the secondary is flooded with valid and signed NOTIFY queries, it is under a replay attack which is discussed in Section 11.5. The key thing here is that the secondary is likely to be designed to be able to process much more traffic than the Hidden Primary hosted on a CPE.

This paragraph details how the secondary may limit the NOTIFY queries. Because the Hidden Primary may be renumbered, the secondary SHOULD NOT perform permanent IP filtering based on IP addresses. In addition, a given secondary may be shared among multiple Hidden Primaries which make filtering rules based on IP harder to set. The time at which a NOTIFY is sent by the Hidden Primary is not predictable. However, a flood of NOTIFY messages may be easily detected, as a NOTIFY originated from a given Homenet Zone is expected to have a very limited number of unique source IP addresses, even when renumbering is occurring. As a result, the secondary, MAY rate limit incoming NOTIFY queries.

On the Hidden Primary side, it is recommended that the Hidden Primary sends a NOTIFY as long as the zone has not been updated by the secondary. Multiple SOA queries may indicate the secondary is under attack.

11.3.3. Reflection Attacks involving the Public Authoritative Servers

Reflection attacks involving the Public Authoritative Server(s) are similar to attacks on any Outsourcing Infrastructure. This is not specific to the architecture described in this document, and thus are considered as out of scope.

In fact, one motivation of the architecture described in this document is to expose the Public Authoritative Server(s) to attacks instead of the CPE, as it is believed that the Public Authoritative Server(s) will be better able to defend itself.

11.4. Flooding Attack

The purpose of flooding attacks is mostly resource exhaustion, where the resource can be bandwidth, memory, or CPU for example.

One goal of the architecture described in this document is to limit the surface of attack on the CPE. This is done by outsourcing the DNS service to the Public Authoritative Server(s). By doing so, the CPE limits its DNS interactions between the Hidden Primary and the Synchronization Server. This limits the number of entities the CPE interacts with as well as the scope of DNS exchanges - NOTIFY, SOA, AXFR, IXFR.

The use of an authenticated channel with SIG(0) or TSIG between the CPE and the Synchronization Server, enables detection of illegitimate DNS queries, so appropriate action may be taken - like dropping the queries. If signatures are validated, then most likely, the CPE is under a replay attack, as detailed in Section 11.5

In order to limit the resource required for authentication, it is recommended to use TSIG that uses symmetric cryptography over SIG(0) that uses asymmetric cryptography.

11.5. Replay Attack

Replay attacks consist of an attacker either resending or delaying a legitimate message that has been sent by an authorized user or process. As the Hidden Primary and the Synchronization Server use an authenticated channel, replay attacks are mostly expected to use forged DNS queries in order to provide valid traffic.

From the perspective of an attacker, using a correctly authenticated DNS query may not be detected as an attack and thus may generate a response. Generating and sending a response consumes more resources than either dropping the query by the defender, or generating the query by the attacker, and thus could be used for resource exhaustion

attacks. In addition, as the authentication is performed at the DNS layer, the source IP address could be impersonated in order to perform a reflection attack.

Section 11.3 details how to mitigate reflection attacks and Section 11.4 details how to mitigate resource exhaustion. Both sections assume a context of DoS with a flood of DNS queries. This section suggests a way to limit the attack surface of replay attacks.

As SIG(0) and TSIG use inception and expiration time, the time frame for replay attack is limited. SIG(0) and TSIG recommends a fudge value of 5 minutes. This value has been set as a compromise between possibly loose time synchronization between devices and the valid lifetime of the message. As a result, better time synchronization policies could reduce the time window of the attack.

12. IANA Considerations

This document has no actions for IANA.

13. Acknowledgment

The authors wish to thank Philippe Lemordant for its contributions on the early versions of the draft; Ole Troan for pointing out issues with the IPv6 routed home concept and placing the scope of this document in a wider picture; Mark Townsley for encouragement and injecting a healthy debate on the merits of the idea; Ulrik de Bie for providing alternative solutions; Paul Mockapetris, Christian Jacquenet, Francis Dupont and Ludovic Eschard for their remarks on CPE and low power devices; Olafur Gudmundsson for clarifying DNSSEC capabilities of small devices; Simon Kelley for its feedback as dnsmasq implementer; Andrew Sullivan, Mark Andrew, Ted Lemon, Mikael Abrahamson, Michael Richardson and Ray Bellis for their feedback on handling different views as well as clarifying the impact of outsourcing the zone signing operation outside the CPE; Mark Andrew and Peter Koch for clarifying the renumbering.

14. References

14.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.

- [RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <<http://www.rfc-editor.org/info/rfc1995>>.
- [RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <<http://www.rfc-editor.org/info/rfc1996>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<http://www.rfc-editor.org/info/rfc2136>>.
- [RFC2142] Crocker, D., "Mailbox Names for Common Services, Roles and Functions", RFC 2142, DOI 10.17487/RFC2142, May 1997, <<http://www.rfc-editor.org/info/rfc2142>>.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <<http://www.rfc-editor.org/info/rfc2308>>.
- [RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <<http://www.rfc-editor.org/info/rfc2845>>.
- [RFC2930] Eastlake 3rd, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, DOI 10.17487/RFC2930, September 2000, <<http://www.rfc-editor.org/info/rfc2930>>.
- [RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <<http://www.rfc-editor.org/info/rfc2931>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<http://www.rfc-editor.org/info/rfc4034>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<http://www.rfc-editor.org/info/rfc4301>>.

- [RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, DOI 10.17487/RFC4555, June 2006, <<http://www.rfc-editor.org/info/rfc4555>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<http://www.rfc-editor.org/info/rfc4960>>.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <<http://www.rfc-editor.org/info/rfc5155>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <<http://www.rfc-editor.org/info/rfc5936>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<http://www.rfc-editor.org/info/rfc6347>>.
- [RFC6644] Evans, D., Droms, R., and S. Jiang, "Rebind Capability in DHCPv6 Reconfigure Messages", RFC 6644, DOI 10.17487/RFC6644, July 2012, <<http://www.rfc-editor.org/info/rfc6644>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<http://www.rfc-editor.org/info/rfc6762>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<http://www.rfc-editor.org/info/rfc6763>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<http://www.rfc-editor.org/info/rfc7296>>.

14.2. Informational References

- [I-D.howard-dnsop-ip6rdns]
Howard, L., "Reverse DNS in IPv6 for Internet Service Providers", draft-howard-dnsop-ip6rdns-00 (work in progress), June 2014.
- [I-D.ietf-homenet-naming-architecture-dhc-options]
Migault, D., Cloetens, W., Griffiths, C., and R. Weber, "DHCP Options for Homenet Naming Architecture", draft-ietf-homenet-naming-architecture-dhc-options-02 (work in progress), May 2015.
- [I-D.ietf-opsec-ipv6-host-scanning]
Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", draft-ietf-opsec-ipv6-host-scanning-08 (work in progress), August 2015.
- [RFC1033] Lottor, M., "Domain Administrators Operations Guide", RFC 1033, DOI 10.17487/RFC1033, November 1987,
<<http://www.rfc-editor.org/info/rfc1033>>.
- [RFC4192] Baker, F., Lear, E., and R. Droms, "Procedures for Renumbering an IPv6 Network without a Flag Day", RFC 4192, DOI 10.17487/RFC4192, September 2005,
<<http://www.rfc-editor.org/info/rfc4192>>.
- [RFC7010] Liu, B., Jiang, S., Carpenter, B., Venaas, S., and W. George, "IPv6 Site Renumbering Gap Analysis", RFC 7010, DOI 10.17487/RFC7010, September 2013,
<<http://www.rfc-editor.org/info/rfc7010>>.
- [RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, DOI 10.17487/RFC7344, September 2014,
<<http://www.rfc-editor.org/info/rfc7344>>.
- [RFC7368] Chown, T., Ed., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", RFC 7368, DOI 10.17487/RFC7368, October 2014,
<<http://www.rfc-editor.org/info/rfc7368>>.
- [RFC7558] Lynn, K., Cheshire, S., Blanchet, M., and D. Migault, "Requirements for Scalable DNS-Based Service Discovery (DNS-SD) / Multicast DNS (mDNS) Extensions", RFC 7558, DOI 10.17487/RFC7558, July 2015,
<<http://www.rfc-editor.org/info/rfc7558>>.

Appendix A. Document Change Log

[RFC Editor: This section is to be removed before publication]

-08

- 1: Clarification of the meaning of CPE. The architecture does not consider a single CPE. The CPE represents multiple functions.

-07:

- 1: Ray Hunter is added as a co-author.

-06:

- 2: Ray Hunter is added in acknowledgment.
- 3: Adding Renumbering section with comments from Dallas meeting
- 4: Replacing Master / Primary - Slave / Secondary

Security Consideration has been updated with Reflection attacks, flooding attacks, and replay attacks.

-05:

*Clarifying on handling different views:

- 1: How the CPE may be involved in the resolution and responds without necessarily requesting the Public Authoritative Server(s) (and eventually the Hidden Primary)
- 2: How to handle local scope resolution that is link-local, site-local and NAT IP addresses as well as Private domain names that the administrator does not want to publish outside the home network.

Adding a Privacy Considerations Section

Clarification on pro/cons outsourcing zone-signing

Documenting how to handle reverse zones

Adding reference to RFC 2308

-04:

*Clarifications on zone signing

- *Rewording

- *Adding section on different views

- *architecture clarifications

-03:

- *Simon's comments taken into consideration

- *Adding SOA, PTR considerations

- *Removing DNSSEC performance paragraphs on low power devices

- *Adding SIG(0) as a mechanism for authenticating the servers

- *Goals clarification: the architecture described in the document 1) does not describe new protocols, and 2) can be adapted to specific cases for advance users.

-02:

- *remove interfaces: "Public Authoritative Server Naming Interface" is replaced by "Public Authoritative Server(s)y(ies)". "Public Authoritative Server Management Interface" is replaced by "Synchronization Server".

-01.3:

- *remove the authoritative / resolver services of the CPE.
Implementation dependent

- *remove interactions with mdns and dhcp. Implementation dependent.

- *remove considerations on low powered devices

- *remove position toward homenet arch

- *remove problem statement section

-01.2:

- * add a CPE description to show that the architecture can fit CPEs

- * specification of the architecture for very low powered devices.

- * integrate mDNS and DHCP interactions with the Homenet Naming Architecture.

* Restructuring the draft. 1) We start from the homenet-arch draft to derive a Naming Architecture, then 2) we show why CPE need mechanisms that do not expose them to the Internet, 3) we describe the mechanisms.

* I remove the terminology and expose it in the figures A and B.

* remove the Front End Homenet Naming Architecture to Homenet Naming
-01:

* Added C. Griffiths as co-author.

* Updated section 5.4 and other sections of draft to update section on Hidden Primary / Slave functions with CPE as Hidden Primary/Homenet Server.

* For next version, address functions of MDNS within Homenet Lan and publishing details northbound via Hidden Primary.

-00: First version published.

Authors' Addresses

Daniel Migault
Ericsson
8400 Boulevard Decarie
Montreal, QC H4P 2N2
Canada

Phone: +1 (514) 452-2160
Email: daniel.migault@ericsson.com

Ralf Weber
Nominum
2000 Seaport Blvd #400
Redwood City, CA 94063
US

Email: ralf.weber@nominum.com
URI: <http://www.nominum.com>

Ray Hunter
Globis Consulting BV
Weegschaalstraat 3
5632CW Eindhoven
The Netherlands

Email: v6ops@globis.net
URI: <http://www.globis.net>

Chris Griffiths

Email: cgriffiths@gmail.com

Wouter Cloetens
SoftAtHome
vaartdijk 3 701
3018 Wijgmaal
Belgium

Email: wouter.cloetens@softathome.com

Homenet Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 30, 2016

M. Stenberg
S. Barth
Independent
P. Pfister
Cisco Systems
November 27, 2015

Home Networking Control Protocol
draft-ietf-homenet-hncp-10

Abstract

This document describes the Home Networking Control Protocol (HNCP), an extensible configuration protocol and a set of requirements for home network devices. HNCP is described as a profile of and extension to the Distributed Node Consensus Protocol (DNCP). HNCP enables discovery of network borders, automated configuration of addresses, name resolution, service discovery, and the use of any routing protocol which supports routing based on both source and destination address.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 30, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Applicability	4
2. Terminology	5
2.1. Requirements language	7
3. DNCP Profile	7
4. HNCP Versioning and Router Capabilities	8
5. Interface Classification	9
5.1. Interface Categories	9
5.2. DHCP Aided Auto-Detection	10
5.3. Algorithm for Border Discovery	10
6. Autonomous Address Configuration	11
6.1. Common Link	12
6.2. External Connections	12
6.3. Prefix Assignment	14
6.3.1. Prefix Assignment Algorithm Parameters	14
6.3.2. Making New Assignments	15
6.3.3. Applying Assignments	16
6.3.4. DHCPv6 Prefix Delegation	16
6.4. Node Address Assignment	17
6.5. Local IPv4 and ULA Prefixes	18
7. Configuration of Hosts and non-HNCP Routers	19
7.1. IPv6 Addressing and Configuration	19
7.2. DHCPv6 for Prefix Delegation	20
7.3. DHCPv4 for Addressing and Configuration	20
7.4. Multicast DNS Proxy	20
8. Naming and Service Discovery	21
9. Securing Third-Party Protocols	22
10. Type-Length-Value Objects	22
10.1. HNCP Version TLV	23
10.2. External Connection TLV	24
10.2.1. Delegated Prefix TLV	24
10.2.2. DHCPv6 Data TLV	26
10.2.3. DHCPv4 Data TLV	26
10.3. Assigned Prefix TLV	27
10.4. Node Address TLV	28
10.5. DNS Delegated Zone TLV	28
10.6. Domain Name TLV	29
10.7. Node Name TLV	30
10.8. Managed PSK TLV	30
11. General Requirements for HNCP Nodes	31

12. Security Considerations	33
12.1. Interface Classification	33
12.2. Security of Unicast Traffic	34
12.3. Other Protocols in the Home	34
13. IANA Considerations	35
14. References	35
14.1. Normative references	35
14.2. Informative references	37
Appendix A. Changelog [RFC Editor: please remove]	38
Appendix B. Draft source [RFC Editor: please remove]	39
Appendix C. Implementation [RFC Editor: please remove]	39
Appendix D. Acknowledgments	40
Authors' Addresses	40

1. Introduction

Home Networking Control Protocol (HNCP) is designed to facilitate sharing of state among home routers to fulfill the needs of the IPv6 homenet architecture [RFC7368], which assumes zero-configuration operation, multiple subnets, multiple home routers and (potentially) multiple upstream service providers providing (potentially) multiple prefixes to the home network. While RFC7368 sets no requirements for IPv4 support, HNCP aims to support dual-stack mode of operation, and therefore the functionality is designed with that in mind. The state is shared as TLVs transported in the DNCP node state among the routers (and potentially advanced hosts) to enable:

- o Autonomic discovery of network borders (Section 5.3) based on Distributed Node Consensus Protocol (DNCP) topology.
- o Automated portioning of prefixes delegated by the service providers as well as assigned prefixes to both HNCP and non-HNCP routers (Section 6.3) using [RFC7695]. Prefixes assigned to HNCP routers are used to:
 - * Provide addresses to non-HNCP aware nodes (using SLAAC and DHCP).
 - * Provide space in which HNCP nodes assign their own addresses (Section 6.4).
- o Internal and external name resolution, as well as multi-link service discovery (Section 8).
- o Other services not defined in this document, that do need to share state among homenet nodes, and do not cause rapid and constant TLV changes (see following applicability section).

HNCP is a DNCP [I-D.ietf-homenet-dncp]-based protocol and includes a DNCP profile which defines transport and synchronization details for sharing state across nodes defined in Section 3. The rest of the document defines behavior of the services noted above, how the required TLVs are encoded (Section 10), as well as additional requirements on how HNCP nodes should behave (Section 11).

1.1. Applicability

While HNCP does not deal with routing protocols directly (except potentially informing them about internal and external interfaces if classification specified in Section 5.3 is used), in homenet environments where multiple IPv6 source-prefixes can be present, routing based on source and destination address is necessary [RFC7368]. Ideally, the routing protocol is also zero-configuration (e.g., no need to configure identifiers or metrics) although HNCP can be used also with a manually configured routing protocol.

As HNCP uses DNCP as the actual state synchronization protocol, the applicability statement of DNCP applies here as well; HNCP should not be used for any data that changes rapidly and constantly. If such data needs to be published in an HNCP network, a more applicable protocol should be used for those portions and locators to a server of said protocol can be announced using HNCP instead. An example for this is naming and service discovery (Section 8) for which HNCP only transports DNS server addresses, and no actual per-name or per-service data of hosts.

HNCP TLVs specified within this document, in steady state, stay constant, with one exception: as Delegated Prefix TLVs (Section 10.2.1) do contain lifetimes, they force re-publishing of that data every time the valid or preferred lifetimes of prefixes are updated (significantly). Therefore, it is desirable for ISPs to provide large enough valid and preferred lifetimes to avoid unnecessary HNCP state churn in homes, but even given non-cooperating ISPs, the state churn is proportional only to the number of externally received delegated prefixes and not the home network size, and should therefore be relatively low.

HNCP assumes a certain level of control over host configuration servers (e.g., DHCP [RFC2131]) on links that are managed by its routers. Some HNCP functionality (such as border discovery or some aspects of naming) might be affected by existing DHCP servers not aware of the HNCP-managed network and thus might need to be reconfigured to not result in unexpected behavior.

While HNCP routers can provide configuration to and receive configuration from non-HNCP routers, they are not able to traverse

such devices based solely on the protocol as defined in this document, i.e., HNCP routers that are connected only by different interfaces of a non-HNCP router will not be part of the same HNCP network.

While HNCP is designed to be used by (home) routers, it can also be used by advanced hosts that want to do, e.g., their own address assignment and routing.

HNCP is link layer agnostic; if a link supports IPv6 (link-local) multicast and unicast, HNCP will work on it. Trickle retransmissions and keep-alives will handle both packet loss and non-transitive connectivity, ensuring eventual convergence.

2. Terminology

The following terms are used as they are defined in [RFC7695]:

- o Advertised Prefix Priority
- o Advertised Prefix
- o Assigned Prefix
- o Delegated Prefix
- o Prefix Adoption
- o Private Link
- o Published Assigned Prefix
- o Applied Assigned Prefix
- o Shared Link

The following terms are used as they are defined in [I-D.ietf-homenet-dncp]:

- o DNCP profile
- o Node identifier
- o Link
- o Interface

(HNCP) node	A device implementing this specification.
(HNCP) router	A device implementing this specification, which forwards traffic on behalf of other devices.
highest node identifier	When comparing the HNCP node identifiers of multiple nodes, the one that has the highest value in a bitwise comparison.
Border	separation point between administrative domains; in this case, between the home network and any other network, i.e., usually an ISP network.
Internal link	a link that does not cross borders.
Internal interface	an interface that is connected to an internal link.
External interface	an interface that is connected to a link which is not an internal link.
Interface category	a local configuration denoting the use of a particular interface. The interface category determines how a HNCP node should treat the particular interface. External and internal category mark the interface as out of or within the network border; there are also a number of sub-categories to internal that further affect local node behavior. See Section 5.1 for a list of interface categories and how they behave. The internal or external categories may also be auto-detected (Section 5.3).
Border router	a router announcing external connectivity and forwarding traffic across the network border.
Common Link	a set of nodes on a link which share a common view of it, i.e., they see each other's traffic and the same set of hosts. Unless configured otherwise transitive connectivity is assumed.
DHCPv4	refers to Dynamic Host Configuration Protocol [RFC2131] in this document.
DHCPv6	refers to Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC3315] in this document.
DHCP	refers to cases which apply to both DHCPv4 and DHCPv6 in this document.

2.1. Requirements language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. DNCP Profile

The DNCP profile for HNCP is defined as follows:

- o HNCP uses UDP datagrams on port HNCP-UDP-PORT as a transport over link-local scoped IPv6, using unicast and multicast (All-Homenet-Nodes is the HNCP group address). Received datagrams where either or both of the IPv6 source or destination address is not link-local scoped MUST be ignored. Replies to multicast and unicast messages MUST be sent to the IPv6 source address and port of the original message. Each node MUST be able to receive (and potentially reassemble) UDP datagrams with a payload of at least 4000 bytes.
- o HNCP operates on multicast-capable interfaces only. HNCP nodes MUST assign a non-zero 32-bit endpoint identifier to each interface for which HNCP is enabled. The value zero is not used in DNCP TLVs, but has a special meaning in HNCP TLVs (see Section 10.3 and Section 6.4). These identifiers MUST be locally unique within the scope of the node and using values equivalent to the IPv6 link-local scope identifiers for the given interfaces are RECOMMENDED.
- o HNCP uses opaque 32-bit node identifiers (DNCP_NODE_IDENTIFIER_LENGTH = 32). A node implementing HNCP SHOULD use a random node identifier. If there is a node identifier collision (as specified in the Node State TLV handling of Section 4.4 of [I-D.ietf-homenet-dncp]), the node MUST immediately generate and use a new random node identifier which is not used by any other node at the time, based on the current DNCP network state.
- o HNCP nodes MUST use the leading 64 bits of the MD5 message digest [RFC1321] as the DNCP hash function H(x) used in building the DNCP hash tree.
- o HNCP nodes MUST use DNCP's per-endpoint keep-alive extension on all endpoints. The following parameters are suggested:
 - * Default keep-alive interval (DNCP_KEEPA_LIVE_INTERVAL): 20 seconds.

- * Multiplier (DNCP_KEEPALIVE_MULTIPLIER): 2.1 on virtually lossless links works fine as it allows for one lost keep-alive. If used on a lossy link, considerably higher multiplier, such as 15, should be used instead. In that case, an implementation might prefer shorter keep-alive intervals on that link as well to ensure that DNCP_KEEPALIVE_INTERVAL * DNCP_KEEPALIVE_MULTIPLIER timeout after which (entirely) lost nodes time out is low enough.
- o HNCP nodes use the following Trickle parameters for the per-interface Trickle instances:
 - * k SHOULD be 1, as the timer reset when data is updated and further retransmissions should handle packet loss. Even on a non-transitive lossy link, the eventual per-endpoint keep-alives should ensure status synchronization occurs.
 - * Imin SHOULD be 200 milliseconds but MUST NOT be lower. Note: Earliest transmissions may occur at Imin / 2.
 - * Imax SHOULD be 7 doublings of Imin [RFC6206] but MUST NOT be lower.
- o HNCP unicast traffic SHOULD be secured using DTLS [RFC6347] as described in DNCP if exchanged over unsecured links. UDP on port HNCP-DTLS-PORT is used for this purpose. A node implementing HNCP security MUST support the DNCP Pre-Shared Key method, SHOULD support the PKI-based trust method and MAY support the DNCP Certificate Based Trust Consensus method. [RFC7525] provides guidance on how to securely utilize DTLS.
- o HNCP nodes MUST ignore all Node State TLVs received via multicast on a link which has DNCP security enabled in order to prevent spoofing of node state changes.

4. HNCP Versioning and Router Capabilities

Multiple versions of HNCP based on compatible DNCP profiles may be present in the same network when transitioning between HNCP versions and for troubleshooting purposes it might be beneficial to identify the HNCP agent version running. Therefore each node MUST include an HNCP-Version TLV (Section 10.1) indicating the currently supported version in its Node Data and MUST ignore (except for DNCP synchronization purposes) any TLVs with a type greater than 32 published by nodes not also publishing an HNCP-Version TLV.

HNCP routers may also have different capabilities regarding interactions with hosts, e.g., for configuration or service

discovery. These are indicated by M, P, H and L values. The combined "capability value" is a metric indicated by interpreting the bits as an integer, i.e., $(M \ll 12 \mid P \ll 8 \mid H \ll 4 \mid L)$. These values are used to elect certain servers on a Common Link, as described in Section 7. Nodes that are not routers MUST announce the value 0 for all capabilities. Any node announcing the value 0 for a capability is considered to not advertise said capability and thus does not take part in the respective election.

5. Interface Classification

5.1. Interface Categories

HNCP specifies the following categories interfaces can be configured to be in:

Internal category: This declares an interfaces to be internal, i.e., within the borders of the HNCP network. The interface MUST operate as a DNCP endpoint. Routers MUST forward traffic with appropriate source addresses between their internal interfaces and allow internal traffic to reach external networks. All nodes MUST implement this category and nodes not implementing any other category implicitly use it as a fixed default.

External category: This declares an interface to be external, i.e., not within the borders of the HNCP network. The interface MUST NOT operate as a DNCP endpoint. Accessing internal resources from external interfaces is restricted, i.e., the use of Recommended Simple Security Capabilities in CPEs [RFC6092] is RECOMMENDED. HNCP routers SHOULD announce acquired configuration information for use in the network as described in Section 6.2, if the interface appears to be connected to an external network. HNCP routers MUST implement this category.

Leaf category: This declares an interface used by client devices only. Such an interface uses the Internal category with the exception that it MUST NOT operate as a DNCP endpoint This category SHOULD be supported by HNCP routers.

Guest category: This declares an interface used by untrusted client devices only. In addition to the restrictions of the Leaf category, HNCP routers MUST filter traffic from and to the interface such that connected devices are unable to reach other devices inside the HNCP network or query services advertised by them unless explicitly allowed. This category SHOULD be supported by HNCP routers.

Ad-hoc category: This configures an interface to use the Internal category but no assumption is made about the the link's transitivity. All other interface categories assume transitive connectivity. This affects the Common Link (Section 6.1) definition. Support for this category is OPTIONAL.

Hybrid category: This declares an interface to use the Internal category while still trying to acquire (external) configuration information on it, e.g., by running DHCP clients. This is useful, e.g., if the link is shared with a non-HNCP router under control and still within the borders of the same network. Detection of this category automatically in addition to manual configuration is out of scope of this document. Support for this category is OPTIONAL.

5.2. DHCP Aided Auto-Detection

Auto-detection of interface categories is possible based on interaction with DHCPv4 [RFC2131] and DHCPv6-PD [RFC3633] servers on connected links. HNCP defines special DHCP behavior to differentiate its internal servers from external ones in order to achieve this. Therefore all internal devices (including HNCP nodes) running DHCP servers on links where auto-detection is used by any HNCP node MUST use the following mechanism based on The User Class Option for DHCPv4 [RFC3004] and its DHCPv6 counterpart [RFC3315]:

- o The device MUST ignore or reject DHCP-Requests containing a DHCP User-Class consisting of the ASCII-String "HOMENET".

Not following this rule (e.g., running unmodified DHCP servers) might lead to false positives when auto-detection is used, i.e., HNCP nodes assume an interface to not be internal, even though it was intended to be.

5.3. Algorithm for Border Discovery

This section defines the interface classification algorithm. It is suitable for both IPv4 and IPv6 (single or dual-stack) and detects the category of an interface either automatically or based on a fixed configuration. By determining the category for all interfaces, the network borders are implicitly defined, i.e., all interfaces not belonging to the External category are considered to be within the borders of the network, all others are not.

The following algorithm MUST be implemented by any node implementing HNCP. However, if the node does not implement auto-detection, only the first and last step are required. The algorithm works as follows, with evaluation stopping at first match:

1. If a fixed category is configured for an interface, it is used.
2. If a delegated prefix could be acquired by running a DHCPv6 client, it is considered external. The DHCPv6 client MUST have included a DHCPv6 User-Class consisting of the ASCII-String "HOMENET" in all of its requests.
3. If an IPv4 address could be acquired by running a DHCPv4 client on the interface, it is considered external. The DHCPv4 client MUST have included a DHCP User-Class consisting of the ASCII-String "HOMENET" in all of its requests.
4. The interface is considered internal.

Note that as other HNCP nodes will ignore the client due to the user class option, any server that replies is clearly external (or a malicious internal node).

An HNCP router SHOULD allow setting the fixed category for each interface which may be connected to either an internal or external device (e.g., an Ethernet port that can be connected to a modem, another HNCP router or a client). Note that all fixed categories except internal and external cannot be auto-detected and can only be selected using manual configuration.

An HNCP router using auto-detection on an interface MUST run the appropriately configured DHCP clients as long as the interface without a fixed category is active (including states where auto-detection considers it to be internal) and rerun the algorithm above to react to conditions resulting in a different interface category. The router SHOULD wait for a reasonable time period (5 seconds as a default), during which the DHCP clients can acquire a lease, before treating a newly activated or previously external interface as internal.

6. Autonomous Address Configuration

This section specifies how HNCP nodes configure host and node addresses. At first border routers share information obtained from service providers or local configuration by publishing one or more External Connection TLVs (Section 10.2). These contain other TLVs such as Delegated Prefix TLVs (Section 10.2.1) which are then used for prefix assignment. Finally, HNCP nodes obtain addresses either statelessly or using a specific stateful mechanism (Section 6.4). Hosts and non-HNCP routers are configured using SLAAC, DHCP or DHCPv6-PD.

6.1. Common Link

HNCP uses the concept of Common Link both in autonomic address configuration and naming and service discovery (Section 8). A Common Link refers to the set of interfaces of nodes that see each other's traffic and presumably also the traffic of all hosts that may use the nodes to, e.g., forward traffic. Common Links are used, e.g., to determine where prefixes should be assigned or which peers participate in the election of a DHCP server. The Common Link is computed separately for each local internal interface, and it always contains the local interface. Additionally, if the local interface is not set to ad-hoc category (see Section 5.1), it also contains the set of interfaces that are bidirectionally reachable from the given local interface, that is, every remote interface of a remote node meeting all of the following requirements:

- o The local node publishes a Peer TLV with:
 - * Peer Node Identifier = remote node's node identifier
 - * Peer Endpoint Identifier = remote interface's endpoint identifier
 - * Endpoint Identifier = local interface's endpoint identifier
- o The remote node publishes a Peer TLV with:
 - * Peer Node Identifier = local node's node identifier
 - * Peer Endpoint Identifier = local interface's endpoint identifier
 - * Endpoint Identifier = remote interface's endpoint identifier

A node MUST be able to detect whether two of its local internal interfaces are connected, e.g., by detecting an identical remote interface being part of the Common Links of both local interfaces.

6.2. External Connections

Each HNCP router MAY obtain external connection information such as address prefixes, DNS server addresses and DNS search paths from one or more sources, e.g., DHCPv6-PD [RFC3633], NETCONF [RFC6241] or static configuration. Each individual external connection to be shared in the network is represented by one External Connection TLV (Section 10.2).

Announcements of individual external connections can consist of the following components:

Delegated Prefixes: address space available for assignment to internal links announced using Delegated Prefix TLVs (Section 10.2.1). Some address spaces might have special properties which are necessary to understand in order to handle them (e.g., information similar to [RFC6603]). This information is encoded using DHCPv6 Data TLVs (Section 10.2.2) inside the respective Delegated Prefix TLVs.

Auxiliary Information: information about services such as DNS or time synchronization regularly used by hosts in addition to addressing and routing information. This information is encoded using DHCPv6 Data TLVs (Section 10.2.2) and DHCPv4 Data TLVs (Section 10.2.3).

Whenever information about reserved parts (e.g., as specified in [RFC6603]) is received for a delegated prefix, the reserved parts **MUST** be advertised using Assigned Prefix TLVs (Section 10.3) with the highest priority (i.e., 15), as if they were assigned to a Private Link.

Some connections or delegated prefixes may have a special meaning and are not regularly used for internal or internet connectivity, instead they may provide access to special services like VPNs, sensor networks, VoIP, IPTV, etc. Care must be taken that these prefixes are properly integrated and dealt with in the network, in order to avoid breaking connectivity for devices who are not aware of their special characteristics or to only selectively allow certain devices to use them. Such prefixes are distinguished using Prefix Policy TLVs (Section 10.2.1.1). Their contents **MAY** be partly opaque to HNCP nodes, and their identification and usage depends on local policy. However the following general rules **MUST** be adhered to:

Special rules apply when making address assignments for prefixes with Prefix Policy TLVs with type 131, as described in Section 6.3.2

In presence of any type 1 to 128 Prefix Policy TLV the prefix is specialized to reach destinations denoted by any such Prefix Policy TLV, i.e., in absence of a type 0 Prefix Policy TLV it is not usable for general internet connectivity. An HNCP router **MAY** enforce this restriction with appropriate packet filter rules.

6.3. Prefix Assignment

HNCP uses the Prefix Assignment Algorithm [RFC7695] in order to assign prefixes to HNCP internal links and uses some of the terminology (Section 2) defined there. HNCP furthermore defines the Assigned Prefix TLV (Section 10.3) which MUST be used to announce Published Assigned Prefixes.

6.3.1. Prefix Assignment Algorithm Parameters

All HNCP nodes running the prefix assignment algorithm use the following values for its parameters:

Node IDs: HNCP node identifiers are used. The comparison operation is defined as bit-wise comparison.

Set of Delegated Prefixes: The set of prefixes encoded in Delegated Prefix TLVs which are not strictly included in prefixes encoded in other Delegated Prefix TLVs. Note that Delegated Prefix TLVs included in ignored External Connection TLVs are not considered. It is dynamically updated as Delegated Prefix TLVs are added or removed.

Set of Shared Links: The set of Common Links associated with interfaces with internal, leaf, guest or ad-hoc category. It is dynamically updated as interfaces are added, removed, or switch from one category to another. When multiple interfaces are detected as belonging to the same Common Link, prefix assignment is disabled on all of these interfaces except one.

Set of Private Links: This document defines Private Links representing DHCPv6-PD clients or as a mean to advertise prefixes included in the DHCPv6 Exclude Prefix option. Other implementation-specific Private Links may be defined whenever a prefix needs to be assigned for a purpose that does not require a consensus with other HNCP nodes.

Set of Advertised Prefixes: The set of prefixes included in Assigned Prefix TLVs advertised by other HNCP nodes (Prefixes advertised by the local node are not in this set). The associated Advertised Prefix Priority is the priority specified in the TLV. The associated Shared Link is determined as follows:

- * If the Link Identifier is zero, the Advertised Prefix is not assigned on a Shared Link.
- * If the other node's interface identified by the Link Identifier is included in one of the Common Links used for prefix

assignment, it is considered as assigned on the given Common Link.

- * Otherwise, the Advertised Prefix is not assigned on a Shared Link.

Advertised Prefixes as well as their associated priorities and associated Shared Links MUST be updated as Assigned Prefix TLVs are added, updated or removed, and as Common Links are modified.

ADOPT_MAX_DELAY: The default value is 0 seconds (i.e., prefix adoption is done instantly).

BACKOFF_MAX_DELAY: The default value is 4 seconds.

RANDOM_SET_SIZE: The default value is 64.

Flooding Delay: The default value is 5 seconds.

Default Advertised Prefix Priority: When a new assignment is created or an assignment is adopted - as specified in the prefix assignment algorithm routine - the default Advertised Prefix Priority to be used is 2.

6.3.2. Making New Assignments

Whenever the prefix assignment algorithm subroutine (Section 4.1 of [RFC7695]) is run on a Common Link and whenever a new prefix may be assigned (case 1 of the subroutine: no Best Assignment and no Current Assignment), the decision of whether the assignment of a new prefix is desired MUST follow these rules in order:

If the Delegated Prefix TLV contains a DHCPv6 Data TLV, and the meaning of one of the DHCP options is not understood by the HNCP node, the creation of a new prefix is not desired. This rule applies to TLVs inside Delegated Prefix TLVs but not to those inside External Connection TLVs.

If the remaining preferred lifetime of the prefix is 0 and there is another delegated prefix of the same IP version used for prefix assignment with a non-zero preferred lifetime, the creation of a new prefix is not desired.

If the Delegated Prefix does not include a Prefix Policy TLV indicating restrictive assignment (type 131) or if local policy exists to identify it based on, e.g., other Prefix Policy TLV values and allows assignment, the creation of a new prefix is desired.

Otherwise, the creation of a new prefix is not desired.

If the considered delegated prefix is an IPv6 prefix, and whenever there is at least one available prefix of length 64, a prefix of length 64 MUST be selected unless configured otherwise. In case no prefix of length 64 would be available, a longer prefix MAY be selected even without configuration.

If the considered delegated prefix is an IPv4 prefix (Section 6.5 details how IPv4 delegated prefixes are generated), a prefix of length 24 SHOULD be preferred.

In any case, an HNCP router making an assignment MUST support a mechanism suitable to distribute addresses from the considered prefix if the link is intended to be used by clients. In this case a router assigning an IPv4 prefix MUST announce the L-capability and a router assigning an IPv6 prefix with a length greater than 64 MUST announce the H-capability as defined in Section 4.

6.3.3. Applying Assignments

The prefix assignment algorithm indicates when a prefix is applied to the respective Common Link. When that happens each router connected to said link:

MUST forward traffic destined to said prefix to the respective link.

MUST participate in the client configuration election as described in Section 7, if the link is intended to be used by clients.

MAY add an address from said prefix to the respective network interface as described in Section 6.4, e.g., if it is to be used as source for locally originating traffic.

6.3.4. DHCPv6 Prefix Delegation

When an HNCP router announcing the P-Capability (Section 4) receives a DHCPv6-PD request from a client, it SHOULD assign one prefix per delegated prefix in the network. This set of assigned prefixes is then delegated to the client, after it has been applied as described in the prefix assignment algorithm. Each DHCPv6-PD client MUST be considered as an independent Private Link and delegation MUST be based on the same set of Delegated Prefixes as the one used for Common Link prefix assignments, however the prefix length to be delegated MAY be smaller than 64.

The assigned prefixes MUST NOT be given to DHCPv6-PD clients before they are applied, and MUST be withdrawn whenever they are destroyed. As an exception to this rule, in order to shorten delays of processed requests, a router MAY prematurely give out a prefix which is advertised but not yet applied if it does so with a valid lifetime of not more than 30 seconds and ensures removal or correction of lifetimes as soon as possible.

6.4. Node Address Assignment

This section specifies how HNCP nodes reserve addresses for their own use. Nodes MAY, at any time, try to reserve a new address from any Applied Assigned Prefix. Each HNCP node SHOULD announce an IPv6 address and - if it supports IPv4 - MUST announce an IPv4 address, whenever matching prefixes are assigned to at least one of its Common Links. These addresses are published using Node Address TLVs and used to locally reach HNCP nodes for other services. Nodes SHOULD NOT create and announce more than one assignment per IP version to avoid cluttering the node data with redundant information unless a special use case requires it.

Stateless assignment based on Semantically Opaque Interface Identifiers [RFC7217] SHOULD be used for address assignment whenever possible (e.g., the prefix length is 64), otherwise (e.g., for IPv4 if supported) the following method MUST be used instead: For any assigned prefix for which stateless assignment is not used, the first quarter of the addresses are reserved for HNCP based address assignments, whereas the last three quarters are left to the DHCP elected router (Section 4 specifies the DHCP server election process). For example, if the prefix 192.0.2.0/24 is assigned and applied to a Common Link, addresses included in 192.0.2.0/26 are reserved for HNCP nodes and the remaining addresses are reserved for the elected DHCPv4 server.

HNCP nodes assign themselves addresses, and then (to ensure eventual lack of conflicting assignments) publish the assignments using the Node Address TLV (Section 10.4).

The process of obtaining addresses is specified as follows:

- o A node MUST NOT start advertising an address if it is already advertised by another node.
- o An assigned address MUST be part of an assigned prefix currently applied on a Common Link which includes the interface specified by the endpoint identifier.

- o An address **MUST NOT** be used unless it has been advertised for at least ADDRESS_APPLY_DELAY consecutive seconds, and is still currently being advertised. The default value for ADDRESS_APPLY_DELAY is 3 seconds.
- o Whenever the same address is advertised by more than one node, all but the one advertised by the node with the highest node identifier **MUST** be removed.

6.5. Local IPv4 and ULA Prefixes

HNCP routers can create a Unique Local Address (ULA) or private IPv4 prefix to enable connectivity between local devices. These prefixes are inserted in HNCP as if they were delegated prefixes of a (virtual) external connection (Section 6.2). The following rules apply:

An HNCP router **SHOULD** create a ULA prefix if there is no other IPv6 prefix with a preferred time greater than 0 in the network. It **MAY** also do so, if there are other delegated IPv6 prefixes, but none of which is locally generated (i.e., without any Prefix Policy TLV) and has a preferred time greater than 0. However, it **MUST NOT** do so otherwise. In case multiple locally generated ULA prefixes are present, only the one published by the node with the highest node identifier is kept among those with a preferred time greater than 0 - if there is any.

An HNCP router **MUST** create a private IPv4 prefix [RFC1918] whenever it wishes to provide IPv4 internet connectivity to the network and no other private IPv4 prefix with internet connectivity currently exists. It **MAY** also enable local IPv4 connectivity by creating a private IPv4 prefix if no IPv4 prefix exists but **MUST NOT** do so otherwise. In case multiple IPv4 prefixes are announced, only the one published by the node with the highest node identifier is kept among those with a Prefix Policy of type 0 - if there is any. The router publishing a prefix with internet connectivity **MUST** forward IPv4 traffic to the internet and perform NAT on behalf of the network as long as it publishes the prefix, other routers in the network **MAY** choose not to.

Creation of such ULA and IPv4 prefixes **MUST** be delayed by a random timespan between 0 and 10 seconds in which the router **MUST** scan for others trying to do the same.

When a new ULA prefix is created, the prefix is selected based on the configuration, using the last non-deprecated ULA prefix, or generated based on [RFC4193].

7. Configuration of Hosts and non-HNCP Routers

HNCP routers need to ensure that hosts and non-HNCP downstream routers on internal links are configured with addresses and routes. Since DHCP clients can usually only bind to one server at a time, a per-link and per-service election takes place.

HNCP routers may have different capabilities for configuring downstream devices and providing naming services. Each router **MUST** therefore indicate its capabilities as specified in Section 4 in order to participate as a candidate in the election.

7.1. IPv6 Addressing and Configuration

In general Stateless Address Autoconfiguration [RFC4861] is used for client configuration for its low overhead and fast renumbering capabilities. Therefore each HNCP router sends Router Advertisements on interfaces which are intended to be used by clients and **MUST** at least include a Prefix Information Option for each Applied Assigned Prefix which it assigned to the respective link in every such advertisement. However, stateful DHCPv6 can be used in addition by administrative choice, to, e.g., collect hostnames and use them to provide naming services or whenever stateless configuration is not applicable.

The designated stateful DHCPv6 server for a Common Link (Section 6.1) is elected based on the capabilities described in Section 4. The winner is the router (connected to the Common Link) advertising the greatest H-capability. In case of a tie, Capability Values (Section 4) are compared, and the router with the greatest value is elected. In case of another tie, the router with the highest node identifier is elected among the routers with tied Capability Values.

The elected router **MUST** serve stateful DHCPv6 and **SHOULD** provide naming services for acquired hostnames as outlined in Section 8, all others nodes **MUST NOT**. Stateful addresses **SHOULD** be assigned in a way not hindering fast renumbering even if the DHCPv6 server or client do not support the DHCPv6 reconfigure mechanism, e.g., by only handing out leases from locally-generated (ULA) prefixes and prefixes with a length different from 64, and by using low renew and rebind times (i.e., not longer than 5 minutes). In case no router was elected, stateful DHCPv6 is not provided. Routers which cease to be elected DHCP servers **SHOULD** - when applicable - invalidate remaining existing bindings in order to trigger client reconfiguration.

7.2. DHCPv6 for Prefix Delegation

The designated DHCPv6 server for prefix-delegation on a Common Link is elected based on the capabilities described in Section 4. The winner is the router (connected to the Common Link) advertising the greatest P-capability. In case of a tie, Capability Values (Section 4) are compared, and the router with the greatest value is elected. In case of another tie, the router with the highest node identifier is elected among the routers with tied Capability Values.

The elected router MUST provide prefix-delegation services [RFC3633] on the given link (and follow the rules in Section 6.3.4), all other nodes MUST NOT.

7.3. DHCPv4 for Addressing and Configuration

The designated DHCPv4 server on a Common Link (Section 6.1) is elected based on the capabilities described in Section 4. The winner is the router (connected to the Common Link) advertising the greatest L-capability. In case of a tie, Capability Values (Section 4) are compared, and the router with the greatest value is elected. In case of another tie, the router with the highest node identifier is elected among the routers with tied Capability Values.

The elected router MUST provide DHCPv4 services on the given link, all other nodes MUST NOT. The elected router MUST provide IP addresses from the pool defined in Section 6.4 and MUST announce itself as router [RFC2132] to clients.

DHCPv4 lifetimes renew and rebind times (T1 and T2) SHOULD be short (i.e., not longer than 5 minutes) in order to provide reasonable response times to changes. Routers which cease to be elected DHCP servers SHOULD - when applicable - invalidate remaining existing bindings in order to trigger client reconfiguration.

7.4. Multicast DNS Proxy

The designated MDNS [RFC6762] proxy on a Common Link is elected based on the capabilities described in Section 4. The winner is the router (connected to the Common Link) advertising the greatest M-capability. In case of a tie, Capability Values (Section 4) are compared, and the router with the greatest value is elected. In case of another tie, the router with the highest node identifier is elected among the routers with tied Capability Values.

The elected router MUST provide an MDNS-proxy on the given link and announce it as described in Section 8.

8. Naming and Service Discovery

Network-wide naming and service discovery can greatly improve the user-friendliness of a network. The following mechanism provides means to setup and delegate naming and service discovery across multiple HNCP routers.

Each HNCP router SHOULD provide and advertise a recursive name resolving server to clients which honors the announcements made in Delegated Zone TLVs (Section 10.5), Domain Name TLVs (Section 10.6) and Node Name TLVs (Section 10.7), i.e., delegate queries to the designated name servers and hand out appropriate A, AAAA and PTR records according to the mentioned TLVs.

Each HNCP router SHOULD provide and announce an auto-generated or user-configured name for each internal Common Link (Section 6.1) for which it is the designated DHCPv4, stateful DHCPv6 server, MDNS proxy, or for which it provides forward or reverse DNS services on behalf of connected devices. This announcement is done using Delegated Zone TLVs (Section 10.5) and MUST be unique in the whole network. In case of a conflict the announcement of the node with the highest node identifier takes precedence and all other nodes MUST cease to announce the conflicting TLV. HNCP routers providing recursive name resolving services MUST use the included DNS server address within the TLV to resolve names belonging to the zone as if there was an NS record.

Each HNCP node SHOULD announce a node name for itself to be easily reachable and MAY announce names on behalf of other devices. Announcements are made using Node Name TLVs (Section 10.7) and the announced names MUST be unique in the whole network. In case of a conflict the announcement of the node with the highest node identifier takes precedence and all other nodes MUST cease to announce the conflicting TLV. HNCP routers providing recursive name resolving services as described above MUST resolve such announced names to their respective IP addresses as if there were corresponding A/AAAA records.

Names and unqualified zones are used in an HNCP network to provide naming and service discovery with local significance. A network-wide zone is appended to all single labels or unqualified zones in order to qualify them. ".home" is the default, however an administrator MAY configure announcing of a Domain Name TLV (Section 10.6) for the network to use a different one. In case multiple are announced, the domain of the node with the greatest node identifier takes precedence.

9. Securing Third-Party Protocols

Pre-shared keys (PSKs) are often required to secure (for example) IGPs and other protocols which lack support for asymmetric security. The following mechanism manages PSKs using HNCP to enable bootstrapping of such third-party protocols. The scheme SHOULD NOT be used unless in conjunction with secured HNCP unicast transport (i.e., DTLS), as transferring the PSK in plain-text anywhere in the network is a potential risk, especially as the originator may not know about security (and use of DNCP security) on all links. The following rules define how such a PSK is managed and used:

- o If no Managed PSK TLV (Section 10.8) is currently being announced, an HNCP node using this mechanism MUST create one after a random delay of 0 to 10 seconds with a 32 bytes long random key and add it to its node data.
- o In case multiple nodes announce such a TLV at the same time, all but the one with the greatest node identifier stop advertising it and adopt the remaining one.
- o The node currently advertising the Managed PSK TLV MUST generate and advertise a new random one whenever an unreachable node is removed from the DNCP topology as described in the Section 4.6 of [I-D.ietf-homenet-dncp].

PSKs for individual protocols SHOULD be derived from the random PSK using a suitable one-way hashing algorithm (e.g., by using HMAC-SHA256 based HKDF [RFC6234] with the particular protocol name in the info field) so that disclosure of any derived key does not impact other users of the managed PSK. Furthermore derived PSKs MUST be updated whenever the managed PSK changes.

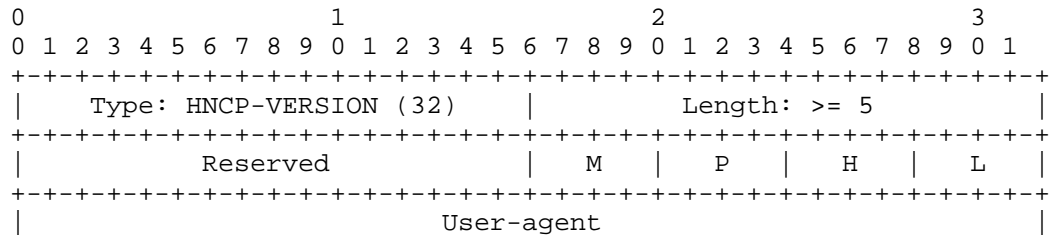
10. Type-Length-Value Objects

HNCP defines the following TLVs in addition to those defined by DNCP. The same general rules and defaults for encoding as noted in Section 7 of [I-D.ietf-homenet-dncp] apply. Note that most HNCP variable-length TLVs also support optional nested TLVs, and they are encoded after the variable length content, followed by the zero padding of the variable length content to the next 32-bit boundary.

TLVs defined here are only valid when appearing in their designated context, i.e., only directly within container TLVs mentioned in their definition, or - absent any mentions - only as top-level TLVs within the node data set. TLVs appearing outside their designated context MUST be ignored.

TLVs encoding IP addresses or prefixes allow encoding both IPv6 and IPv4 addresses and prefixes. IPv6 information is encoded as is, whereas for IPv4 IPv4-mapped IPv6 addresses format [RFC4291] is used and prefix lengths are encoded as original IPv4 prefix length increased by 96.

10.1. HNCP Version TLV



This TLV is used to indicate the supported version and router capabilities of an HNCP node as described in Section 4.

Reserved: Bits are reserved for future use. They MUST be set to zero when creating this TLV, and their value MUST be ignored when processing the TLV.

M-capability: Priority value used for electing the on-link MDNS [RFC6762] proxy. It MUST be set to 0 if the router is not capable of proxying MDNS, otherwise it SHOULD be set to 4 but MAY be set to any value from 1 to 7 to indicate a non-default priority. The values 8-15 are reserved for future use.

P-capability: Priority value used for electing the on-link DHCPv6-PD server. It MUST be set to 0 if the router is not capable of providing prefixes through DHCPv6-PD (Section 6.3.4), otherwise it SHOULD be set to 4 but MAY be set to any value from 1 to 7 to indicate a non-default priority. The values 8-15 are reserved for future use.

H-capability: Priority value used for electing the on-link DHCPv6 server offering non-temporary addresses. It MUST be set to 0 if the router is not capable of providing such addresses, otherwise it SHOULD be set to 4 but MAY be set to any value from 1 to 7 to indicate a non-default priority. The values 8-15 are reserved for future use.

L-capability: Priority value used for electing the on-link DHCPv4 server. It MUST be set to 0 if the router is not capable of running a legacy DHCPv4 server offering IPv4 addresses to clients, otherwise it SHOULD be set to 4 but MAY be set to any value from 1

to 7 to indicate a non-default priority. The values 8-15 are reserved for future use.

User-Agent: The user-agent is a human-readable UTF-8 string that describes the name and version of the current HNCP implementation.

10.2. External Connection TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Type: EXTERNAL-CONNECTION (33) | Length |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               (Optional nested TLVs) |

```

An External Connection TLV is a container TLV used to gather network configuration information associated with a single external connection (Section 6.2) to be shared across the HNCP network. A node MAY publish an arbitrary number of instances of this TLV to share the desired number of external connections. Upon reception, the information transmitted in any nested TLVs is used for the purposes of prefix assignment (Section 6.3) and host configuration (Section 7).

10.2.1. Delegated Prefix TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Type: DELEGATED-PREFIX (34) | Length: >= 9 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Valid Lifetime Since Origination |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Preferred Lifetime Since Origination |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Prefix Length |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Prefix |
...
|                               | 0-pad if any |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               (Optional nested TLVs) |

```

The Delegated Prefix TLV is used by HNCP routers to advertise prefixes which are allocated to the whole network and can be used for prefix assignment. Delegated Prefix TLVs are only valid inside External Connection TLVs and their prefixes MUST NOT overlap with those of other such TLVs in the same container.

Valid Lifetime Since Origination: The time in seconds the delegated prefix was valid for at the origination time of the node data containing this TLV. The value **MUST** be updated whenever the node republishes its Node State TLV.

Preferred Lifetime Since Origination: The time in seconds the delegated prefix was preferred for at the origination time of the node data containing this TLV. The value **MUST** be updated whenever the node republishes its Node State TLV.

Prefix Length: The number of significant bits in the Prefix.

Prefix: Significant bits of the prefix padded with zeroes up to the next byte boundary.

10.2.1.1. Prefix Policy TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type: PREFIX-POLICY (43)   |   Length: >= 1   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Policy Type |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Value                                     +
|

```

The Prefix Policy TLV contains information about the policy or applicability of a delegated prefix. This information can be used to determine whether prefixes for a certain usecase (e.g., local reachability, internet connectivity) do exist or are to be acquired and to make decisions about assigning prefixes to certain links or to fine-tune border firewalls. See Section 6.2 for a more in-depth discussion. This TLV is only valid inside a Delegated Prefix TLV.

Policy Type: The type of the policy identifier.

0 : Internet connectivity (no Value).

1-128 : Explicit destination prefix with the Policy Type being the actual length of the prefix and the Value containing significant bits of the destination prefix padded with zeroes up to the next byte boundary.

129 : DNS Domain. The Value contains an RFC 1035 [RFC1035] encoded DNS label sequence. Compression **MUST NOT** be used. The label sequence **MUST** end with an empty label.

130 : Opaque UTF-8 string (e.g., for administrative purposes).

131 : Restrictive Assignment (no Value).

132-255: Reserved for future additions.

Value: A variable length identifier of the given type.

10.2.2. DHCPv6 Data TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Type: DHCPV6-DATA (37)   |           Length: > 0           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               DHCPv6 option stream            |

```

This TLV is used to encode auxiliary IPv6 configuration information (e.g., recursive DNS servers) encoded as a stream of DHCPv6 options. It is only valid in an External Connection TLV or a Delegated Prefix TLV encoding an IPv6 prefix and MUST NOT occur more than once in any single container. When included in an External Connection TLV, it contains DHCPv6 options relevant to the External Connection as a whole. When included in a Delegated Prefix, it contains options mandatory to handle said prefix.

DHCPv6 option stream: DHCPv6 options encoded as specified in [RFC3315].

10.2.3. DHCPv4 Data TLV

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Type: DHCPV4-DATA (38)   |           Length: > 0           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               DHCPv4 option stream            |

```

This TLV is used to encode auxiliary IPv4 configuration information (e.g., recursive DNS servers) encoded as a stream of DHCPv4 options. It is only valid in an External Connection TLV and MUST NOT occur more than once in any single container. It contains DHCPv4 options relevant to the External Connection as a whole.

DHCPv4 option stream: DHCPv4 options encoded as specified in [RFC2131].

10.3. Assigned Prefix TLV

```

      0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  Type: ASSIGNED-PREFIX (35)  |      Length: >= 6      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Endpoint Identifier       |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  Rsv. | Prty. | Prefix Length |                          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Prefix                      |
...
|                               | 0-pad if any            |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               (Optional nested TLVs)     |

```

This TLV is used to announce Published Assigned Prefixes for the purposes of prefix assignment (Section 6.3).

Endpoint Identifier: The endpoint identifier of the local interface the prefix is assigned to, or 0 if it is assigned to a Private Link (e.g., when the prefix is assigned for downstream prefix delegation).

Rsv.: Bits are reserved for future use. They MUST be set to zero when creating this TLV, and their value MUST be ignored when processing the TLV.

Prty: The Advertised Prefix Priority from 0 to 15.

0-1 : Low priorities.

2 : Default priority.

3-7 : High priorities.

8-11 : Administrative priorities. MUST NOT be used unless configured otherwise.

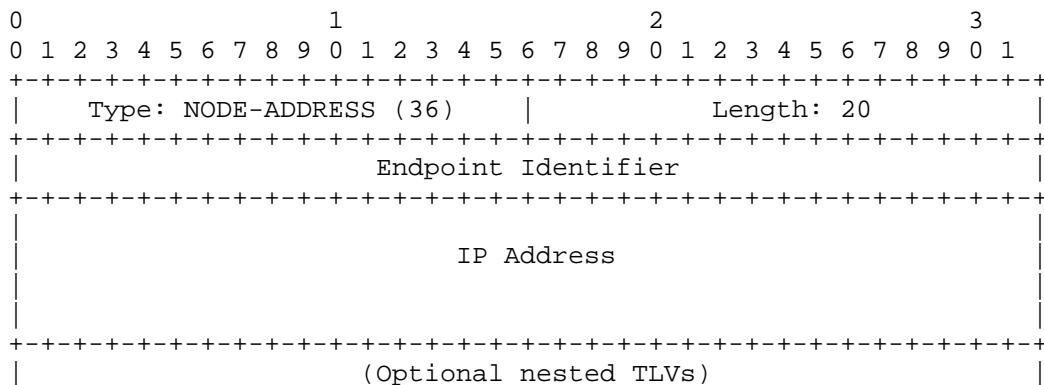
12-14: Reserved for future use.

15 : Provider priorities. MAY only be used by the router advertising the corresponding delegated prefix and based on static or dynamic configuration (e.g., for excluding a prefix based on DHCPv6-PD Prefix Exclude Option [RFC6603]).

Prefix Length: The number of significant bits in the Prefix field.

Prefix: The significant bits of the prefix padded with zeroes up to the next byte boundary.

10.4. Node Address TLV

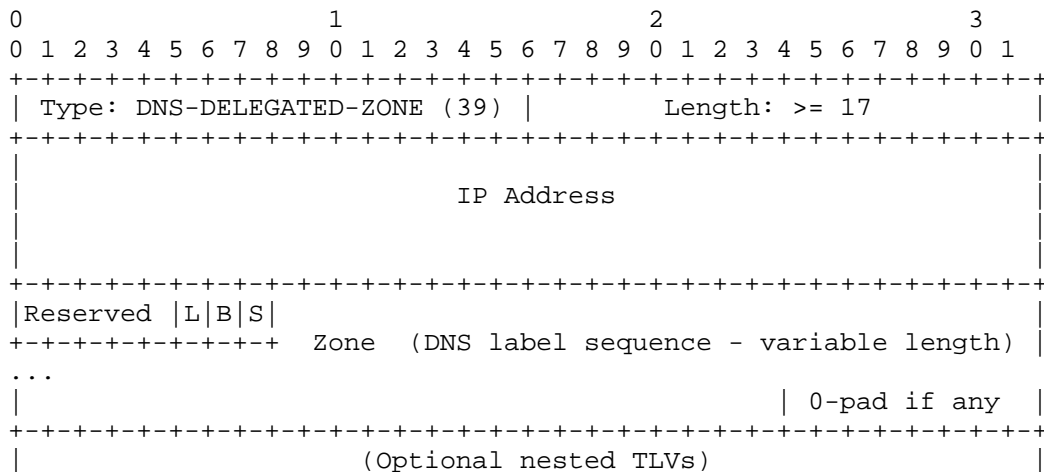


This TLV is used to announce addresses assigned to an HNCP node as described in Section 6.4.

Endpoint Identifier: The endpoint identifier of the local interface the prefix is assigned to, or 0 if it is not assigned on an HNCP enabled link.

IP Address: The globally scoped IPv6 address, or the IPv4 address encoded as an IPv4-mapped IPv6 address [RFC4291].

10.5. DNS Delegated Zone TLV



This TLV is used to announce a forward or reverse DNS zone delegation in the HNCP network. Its meaning is roughly equivalent to specifying an NS and A/AAAA record for said zone. Details are specified in Section 8.

IP Address : The IPv6 address of the authoritative DNS server for the zone; IPv4 addresses are represented as IPv4-mapped addresses [RFC4291]. The special value of :: (all-zero) means the delegation is available in the global DNS-hierarchy.

Reserved : Those bits MUST be set to zero when creating the TLV and ignored when parsing it unless defined in a later specification.

L-bit : DNS-SD [RFC6763] Legacy-Browse, indicates that this delegated zone SHOULD be included in the network's DNS-SD legacy browse list of domains at lb._dns-sd._udp.(DOMAIN-NAME). Local forward zones SHOULD have this bit set, reverse zones SHOULD NOT.

B-bit : (DNS-SD [RFC6763] Browse) indicates that this delegated zone SHOULD be included in the network's DNS-SD browse list of domains at b._dns-sd._udp. (DOMAIN-NAME). Local forward zones SHOULD have this bit set, reverse zones SHOULD NOT.

S-bit : (fully-qualified DNS-SD [RFC6763] domain) indicates that this delegated zone consists of a fully-qualified DNS-SD domain, which should be used as base for DNS-SD domain enumeration, i.e., _dns-sd._udp.(Zone) exists. Forward zones MAY have this bit set, reverse zones MUST NOT. This can be used to provision DNS search path to hosts for non-local services (such as those provided by an ISP, or other manually configured service providers). Zones with this flag SHOULD be added to the search domains advertised to clients.

Zone : The label sequence encoded according to [RFC1035]. Compression MUST NOT be used. The label sequence MUST end with an empty label.

10.6. Domain Name TLV

```

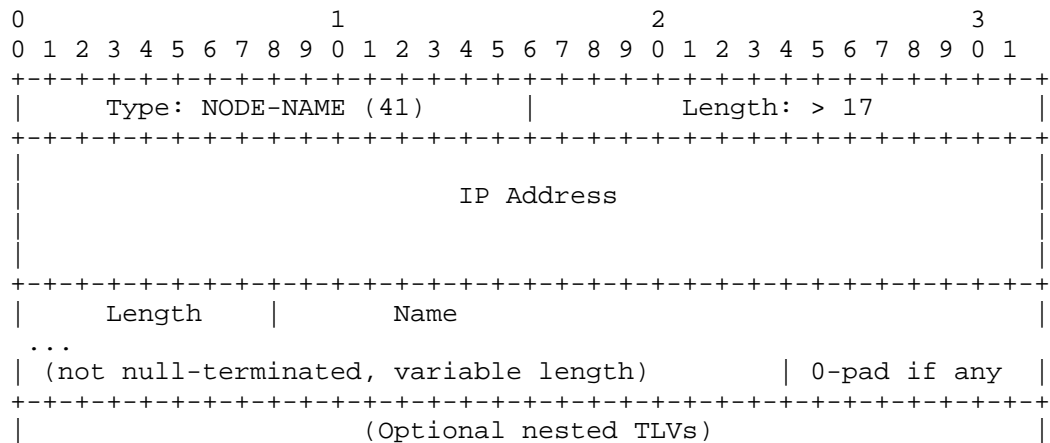
0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type: DOMAIN-NAME (40)   |   Length: > 0   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Domain (DNS label sequence - variable length)   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```


This TLV is used to indicate the base domain name for the network as specified in Section 8. This TLV MUST NOT be announced unless the domain name was explicitly configured by an administrator.

Domain: The label sequence encoded according to [RFC1035].
 Compression MUST NOT be used. The label sequence MUST end with an empty label.

10.7. Node Name TLV



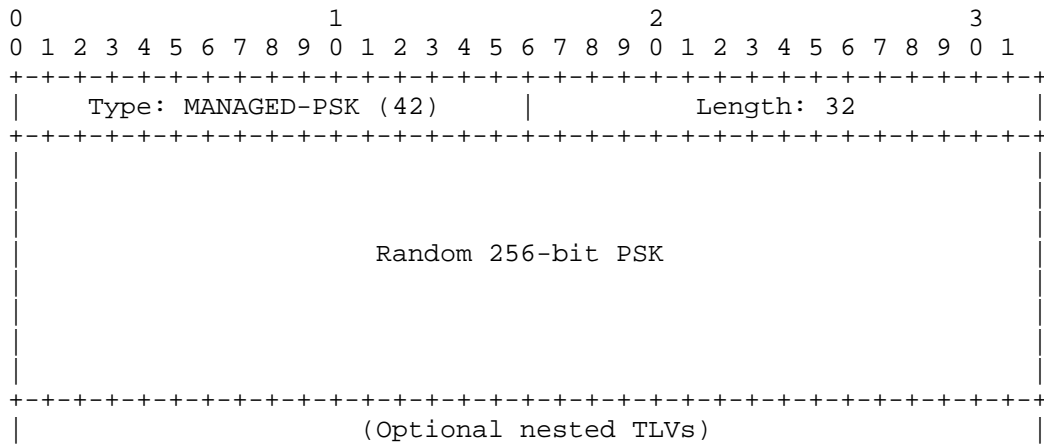
This TLV is used to assign the name of a node in the network to a certain IP address as specified in Section 8.

IP Address: The IP address associated with the name. IPv4 addresses are encoded using IPv4-mapped IPv6 addresses.

Length: The length of the name (0-63).

Name: The name of the node as a single DNS label.

10.8. Managed PSK TLV



This TLV is used to announce a PSK for securing third-party protocols exclusively supporting symmetric cryptography as specified in Section 9.

11. General Requirements for HNCP Nodes

Each node implementing HNCP is subject to the following requirements:

- o It MUST implement HNCP-Versioning (Section 4) and Interface Classification (Section 5).
- o It MUST implement and run the method for securing third-party protocols (Section 9) whenever it uses the security mechanism of HNCP.

If the node is acting as a router, then the following requirements apply in addition:

- o It MUST support Autonomous Address Configuration (Section 6) and Configuration of Hosts and non-HNCP Routers (Section 7).
- o It SHOULD implement support for the Service Discovery and Naming (Section 8) as defined in this document.
- o It MAY be able to provide connectivity to IPv4-devices using DHCPv4.
- o It SHOULD be able to delegate prefixes to legacy IPv6 routers using DHCPv6-PD (Section 6.3.4).

- o In addition, normative language of Basic Requirements for IPv6 Customer Edge Routers [RFC7084] applies with the following adjustments:
 - * The generic requirements G-4 and G-5 are relaxed such that any known default router on any interface is sufficient for a router to announce itself as default router, similarly only the loss of all such default routers results in self-invalidation.
 - * The section "WAN-Side Configuration" applies to interfaces classified as external.
 - * If the CE sends a size-hint as indicated in WPD-2, the hint MUST NOT be determined by the number of LAN-interfaces of the CE, but SHOULD instead be large enough to at least accommodate prefix assignments announced for existing delegated or ULA-prefixes, if such prefixes exist and unless explicitly configured otherwise.
 - * The dropping of packets with a destination address belonging to a delegated prefix mandated in WPD-5 MUST NOT be applied to destinations that are part of any prefix announced using an Assigned Prefix TLV by any HNCP router in the network.
 - * The section "LAN-Side Configuration" applies to interfaces not classified as external.
 - * The requirement L-2 to assign a separate /64 to each LAN interface is replaced by the participation in the prefix assignment mechanism (Section 6.3) for each such interface.
 - * The requirement L-9 is modified, in that the M flag MUST be set if and only if a router connected to the respective Common Link is advertising a non-zero H-capability. The O flag SHOULD always be set.
 - * The requirement L-12 to make DHCPv6 options available is adapted, in that a CER SHOULD publish the subset of options using the DHCPv6 Data TLV in an External Connection TLV. Similarly it SHOULD do the same for DHCPv4 options in a DHCPv4 Data TLV. DHCPv6 options received inside an OPTION_IAPREFIX [RFC3633] MUST be published using a DHCPv6 Data TLV inside the respective Delegated Prefix TLV. HNCP routers SHOULD make relevant DHCPv6 and DHCPv4 options available to clients, i.e., options contained in External Connection TLVs that also include delegated prefixes from which a subset is assigned to the respective link.

- * The requirement L-13 to deprecate prefixes is applied to all delegated prefixes in the network from which assignments have been made on the respective interface. Furthermore the Prefix Information Options indicating deprecation MUST be included in Router Advertisements for the remainder of the prefixes' respective valid lifetime, but MAY be omitted after at least 2 hours have passed.

12. Security Considerations

HNCP enables self-configuring networks, requiring as little user intervention as possible. However this zero-configuration goal usually conflicts with security goals and introduces a number of threats.

General security issues for existing home networks are discussed in [RFC7368]. The protocols used to set up addresses and routes in such networks to this day rarely have security enabled within the configuration protocol itself. However these issues are out of scope for the security of HNCP itself.

HNCP is a DNCP-based state synchronization mechanism carrying information with varying threat potential. For this consideration the payloads defined in DNCP and this document are reviewed:

- o Network topology information such as HNCP nodes and their common links.
- o Address assignment information such as delegated and assigned prefixes for individual links.
- o Naming and service discovery information such as auto-generated or customized names for individual links and nodes.

12.1. Interface Classification

As described in Section 5.3, an HNCP node determines the internal or external state on a per-interface basis. A firewall perimeter is set up for the external interfaces, and for internal interfaces, HNCP traffic is allowed, with the exception of leaf and guest sub-categories.

Threats concerning automatic interface classification cannot be mitigated by encrypting or authenticating HNCP traffic itself since external routers do not participate in the protocol and often cannot be authenticated by other means. These threats include propagation of forged uplinks in the homenet in order to, e.g., redirect traffic

destined to external locations and forged internal status by external routers to, e.g., circumvent the perimeter firewall.

It is therefore imperative to either secure individual links on the physical or link-layer or preconfigure the adjacent interfaces of HNCP routers to an appropriate fixed category in order to secure the homenet border. Depending on the security of the external link eavesdropping, man-in-the-middle and similar attacks on external traffic can still happen between a homenet border router and the ISP, however these cannot be mitigated from inside the homenet. For example, DHCPv4 has defined [RFC3118] to authenticate DHCPv4 messages, but this is very rarely implemented in large or small networks. Further, while PPP can provide secure authentication of both sides of a point to point link, it is most often deployed with one-way authentication of the subscriber to the ISP, not the ISP to the subscriber.

12.2. Security of Unicast Traffic

Once the homenet border has been established there are several ways to secure HNCP against internal threats like manipulation or eavesdropping by compromised devices on a link which is enabled for HNCP traffic. If left unsecured, attackers may perform arbitrary traffic redirection, eavesdropping, spoofing or denial of service attacks on HNCP services such as address assignment or service discovery, and the protocols secured using HNCP-derived keys such as routing protocols.

Detailed interface categories like "leaf" or "guest" can be used to integrate not fully trusted devices to various degrees into the homenet by not exposing them to HNCP traffic or by using firewall rules to prevent them from reaching homenet-internal resources.

On links where this is not practical and lower layers do not provide adequate protection from attackers, DTLS-based secure unicast transport MUST be used to secure traffic.

12.3. Other Protocols in the Home

IGPs and other protocols are usually run alongside HNCP therefore the individual security aspects of the respective protocols must be considered. It can however be summarized that many protocols to be run in the home (like IGPs) provide - to a certain extent - similar security mechanisms. Most of these protocols do not support encryption and only support authentication based on pre-shared keys natively. This influences the effectiveness of any encryption-based security mechanism deployed by HNCP as homenet routing information is thus usually not encrypted.

13. IANA Considerations

IANA should set up a registry for the (decimal values within range 32-511) "HNCP TLV Types" under "Distributed Node Consensus Protocol (DNCP)", with the following initial contents:

- 32: HNCP-Version
- 33: External-Connection
- 34: Delegated-Prefix
- 35: Assigned-Prefix
- 36: Node-Address
- 37: DHCPv4-Data
- 38: DHCPv6-Data
- 39: DNS-Delegated-Zone
- 40: Domain-Name
- 41: Node-Name
- 42: Managed-PSK
- 43: Prefix-Policy
- 44-511: Free - policy of 'RFC required' [RFC5226] should be used.

The range reserved by DNCP for Private Use (768-1023) is used by HNCP for per-implementation experimentation. How collisions are avoided is out of the scope of this document.

HNCP requires allocation of well-known UDP port numbers HNCP-UDP-PORT (service name: hncp-udp-port, description: HNCP) and HNCP-DTLS-PORT (service name: hncp-dtls-port, description: HNCP over DTLS), as well as an IPv6 link-local multicast address All-Homenet-Nodes.

14. References

14.1. Normative references

- [I-D.ietf-homenet-dncp]
Stenberg, M. and S. Barth, "Distributed Node Consensus Protocol", draft-ietf-homenet-dncp-12 (work in progress), November 2015.
- [RFC7695] Pfister, P., Paterson, B., and J. Arkko, "Distributed Prefix Assignment Algorithm", RFC 7695, DOI 10.17487/RFC7695, November 2015, <<http://www.rfc-editor.org/info/rfc7695>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<http://www.rfc-editor.org/info/rfc6347>>.
- [RFC6603] Korhonen, J., Ed., Savolainen, T., Krishnan, S., and O. Troan, "Prefix Exclude Option for DHCPv6-based Prefix Delegation", RFC 6603, DOI 10.17487/RFC6603, May 2012, <<http://www.rfc-editor.org/info/rfc6603>>.
- [RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206, March 2011, <<http://www.rfc-editor.org/info/rfc6206>>.
- [RFC3004] Stump, G., Droms, R., Gu, Y., Vyaghrapuri, R., Demirtjis, A., Beser, B., and J. Privat, "The User Class Option for DHCP", RFC 3004, DOI 10.17487/RFC3004, November 2000, <<http://www.rfc-editor.org/info/rfc3004>>.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<http://www.rfc-editor.org/info/rfc2131>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<http://www.rfc-editor.org/info/rfc3315>>.

- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <<http://www.rfc-editor.org/info/rfc3633>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<http://www.rfc-editor.org/info/rfc4291>>.
- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <<http://www.rfc-editor.org/info/rfc1321>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<http://www.rfc-editor.org/info/rfc6763>>.
- [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997, <<http://www.rfc-editor.org/info/rfc2132>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<http://www.rfc-editor.org/info/rfc4193>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<http://www.rfc-editor.org/info/rfc7217>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.
- [RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <<http://www.rfc-editor.org/info/rfc6092>>.

14.2. Informative references

- [RFC3118] Droms, R. and W. Arbaugh., Ed., "Authentication for DHCP Messages", RFC 3118, DOI 10.17487/RFC3118, June 2001, <<http://www.rfc-editor.org/info/rfc3118>>.

- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <<http://www.rfc-editor.org/info/rfc1918>>.
- [RFC7368] Chown, T., Ed., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", RFC 7368, DOI 10.17487/RFC7368, October 2014, <<http://www.rfc-editor.org/info/rfc7368>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<http://www.rfc-editor.org/info/rfc6234>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<http://www.rfc-editor.org/info/rfc6762>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<http://www.rfc-editor.org/info/rfc6241>>.
- [RFC7084] Singh, H., Beebe, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, DOI 10.17487/RFC7084, November 2013, <<http://www.rfc-editor.org/info/rfc7084>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<http://www.rfc-editor.org/info/rfc7525>>.

Appendix A. Changelog [RFC Editor: please remove]

draft-ietf-homenet-hncp-10: Mainly IESG review based changes, no real content change.

draft-ietf-homenet-hncp-09: Added nested TLV definitions for variable length TLVs. NOTE: Node name TLV encoding includes now length byte. Version TLV now itself indicates version.

draft-ietf-homenet-hncp-08: Editorial reorganization.

draft-ietf-homenet-hncp-07: Using version 1 instead of version 0, as existing implementations already use it.

draft-ietf-homenet-hncp-06: Various edits based on feedback, hopefully without functional delta.

draft-ietf-homenet-hncp-05: Renamed "Adjacent Link" to "Common Link". Changed single IPv4 uplink election from MUST to MAY. Added explicit indication to distinguish (IPv4)-PDs for local connectivity and ones with uplink connectivity allowing, e.g., better local-only IPv4-connectivity.

draft-ietf-homenet-hncp-04: Change the responsibility for sending RAs to the router assigning the prefix.

draft-ietf-homenet-hncp-03: Split to DNCP (generic protocol) and HNCP (homenet profile).

draft-ietf-homenet-hncp-02: Removed any built-in security. Relying on IPsec. Reorganized interface categories, added requirements languages, made manual border configuration a MUST-support. Redesigned routing protocol election to consider non-router devices.

draft-ietf-homenet-hncp-01: Added (MAY) guest, ad-hoc, hybrid categories for interfaces. Removed old hnetv2 reference, and now pointing just to OpenWrt + github. Fixed synchronization algorithm to spread also same update number, but different data hash case. Made purge step require bidirectional connectivity between nodes when traversing the graph. Edited few other things to be hopefully slightly clearer without changing their meaning.

draft-ietf-homenet-hncp-00: Added version TLV to allow for TLV content changes pre-RFC without changing IDs. Added link id to assigned address TLV.

Appendix B. Draft source [RFC Editor: please remove]

This draft is available at <https://github.com/fingon/ietf-drafts/> in source format. Issues and pull requests are welcome.

Appendix C. Implementation [RFC Editor: please remove]

A GPLv2-licensed implementation of HNCP is currently under development at <https://github.com/sbyx/hnetd/> and binaries are available in the OpenWrt package repositories (<http://www.openwrt.org>). See <http://www.homewrt.org/>

`doku.php?id=run-conf` for more information. Feedback and contributions are welcome.

Appendix D. Acknowledgments

Thanks to Ole Troan, Mark Baugher, Mark Townsley, Juliusz Chroboczek and Thomas Clausen for their contributions to the draft.

Thanks to Eric Kline for the original border discovery work.

Authors' Addresses

Markus Stenberg
Independent
Helsinki 00930
Finland

Email: `markus.stenberg@iki.fi`

Steven Barth
Independent
Halle 06114
Germany

Email: `cyrus@openwrt.org`

Pierre Pfister
Cisco Systems
Paris
France

Email: `pierre.pfister@darou.fr`

Homenet Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 17, 2016

M. Stenberg
Independent
October 15, 2015

Auto-Configuration of a Network of Hybrid Unicast/Multicast DNS-Based
Service Discovery Proxy Nodes
draft-ietf-homenet-hybrid-proxy-zeroconf-02

Abstract

This document describes how a proxy functioning between Unicast DNS-Based Service Discovery and Multicast DNS can be automatically configured using an arbitrary network-level state sharing mechanism.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 17, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements language	3
3. Hybrid proxy - what to configure	3
3.1. Conflict resolution within network	4
3.2. Per-link DNS-SD forward zone names	4
3.3. Reasonable defaults	5
3.3.1. Network-wide unique link name (scheme 1)	5
3.3.2. Node name (scheme 2)	5
3.3.3. Link name (scheme 2)	5
4. TLVs	5
4.1. DNS Delegated Zone TLV	6
4.2. Domain Name TLV	7
4.3. Node Name TLV	7
5. Desirable behavior	7
5.1. DNS search path in DHCP requests	8
5.2. Hybrid proxy	8
5.3. Hybrid proxy network zeroconf daemon	8
6. Limited zone stitching for host name resolution	8
7. Security Considerations	9
8. IANA Considerations	9
9. References	9
9.1. Normative references	9
9.2. Informative references	10
Appendix A. Example configuration	10
A.1. Used topology	10
A.2. Zero-configuration steps	11
A.3. TLV state	11
A.4. DNS zone	12
A.5. Interaction with hosts	13
Appendix B. Implementation	13
Appendix C. Why not just proxy Multicast DNS?	13
C.1. General problems	14
C.2. Stateless proxying problems	14
C.3. Stateful proxying problems	15
Appendix D. Acknowledgements	15
Appendix E. Changelog [RFC Editor: please remove]	15
Author's Address	16

1. Introduction

Section 3 ("Hybrid Proxy Operation") of [I-D.ietf-dnssd-hybrid] describes how to translate queries from Unicast DNS-Based Service Discovery described in [RFC6763] to Multicast DNS described in [RFC6762], and how to filter the responses and translate them back to unicast DNS.

This document describes what sort of configuration the participating hybrid proxy servers require, as well as how it can be provided using any network-wide state sharing mechanism such as link-state routing protocol or Home Networking Control Protocol [I-D.ietf-homenet-hncp]. The document also describes a naming scheme which does not even need to be same across the whole covered network to work as long as the specified conflict resolution works. The scheme can be used to provision both forward and reverse DNS zones which employ hybrid proxy for heavy lifting.

This document does not go into low level encoding details of the Type-Length-Value (TLV) data that we want synchronized across a network. Instead, we just specify what needs to be available, and assume every node that needs it has it available.

We go through the mandatory specification of the language used in Section 2, then describe what needs to be configured in hybrid proxies and participating DNS servers across the network in Section 3. How the data is exchanged using arbitrary TLVs is described in Section 4. Finally, some overall notes on desired behavior of different software components is mentioned in Section 5.

2. Requirements language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Hybrid proxy - what to configure

Beyond the low-level translation mechanism between unicast and multicast service discovery, the hybrid proxy draft [I-D.ietf-dnssd-hybrid] describes just that there have to be NS records pointing to hybrid proxy responsible for each link within the covered network.

In zero-configuration case, choosing the links to be covered is also non-trivial choice; we can use the border discovery functionality (if available) to determine internal and external links. Or we can use some other protocol's presence (or lack of it) on a link to determine internal links within the covered network, and some other signs (depending on the deployment) such as DHCPv6 Prefix Delegation (as described in [RFC3633]) to determine external links that should not be covered.

For each covered link we want forward DNS zone delegation to an appropriate node which is connected to a link, and running hybrid proxy. Therefore the links' forward DNS zone names should be unique

across the network. We also want to populate reverse DNS zone similarly for each IPv4 or IPv6 prefix in use.

There should be DNS-SD browse domain list provided for the network's domain which contains each physical link only once, regardless of how many nodes and hybrid proxy implementations are connected to it.

Yet another case to consider is the list of DNS-SD domains that we want hosts to enumerate for browse domain lists. Typically, it contains only the local network's domain, but there may be also other networks we may want to pretend to be local but are in different scope, or controlled by different organization. For example, a home user might see both home domain's services (TBD-TLD), as well as ISP's services under `isp.example.com`.

3.1. Conflict resolution within network

Any naming-related choice on node may have conflicts in the network given that we require only distributed loosely synchronized database. We assume only that the underlying protocol used for synchronization has some concept of precedence between nodes originating conflicting information, and in case of conflict, the higher precedence node MUST keep the name they have chosen. The one(s) with lower precedence MUST either try different one (that is not in use at all according to the current link state information), or choose not to publish the name altogether.

If a node needs to pick a different name, any algorithm works, although simple algorithm choice is just like the one described in Multicast DNS[RFC6762]: append -2, -3, and so forth, until there are no conflicts in the network for the given name.

3.2. Per-link DNS-SD forward zone names

How to name the links of a whole network in automated fashion? Two different approaches seem obvious:

1. Unique link name based - `(unique-link).(domain)`.
2. Node and link name - `(link).(unique-node).(domain)`.

The first choice is appealing as it can be much more friendly (especially given manual configuration). For example, it could mean just `lan.example.com` and `wlan.example.com` for a simple home network. The second choice, on the other hand, has a nice property of being local choice as long as node name can be made unique.

The type of naming scheme to use can be left as implementation option. And the actual names themselves SHOULD be also overridable, if the end-user wants to customize them in some way.

3.3. Reasonable defaults

Note that any manual configuration, which SHOULD be possible, MUST override the defaults provided here or chosen by the creator of the implementation.

3.3.1. Network-wide unique link name (scheme 1)

It is not obvious how to produce network-wide unique link names for the (unique-link).(domain) scheme. One option would be to base it on type of physical network layer, and then hope that the number of the networks won't be significant enough to confuse (e.g. "lan", or "wlan").

The network-wide unique link names should be only used in small networks. Given a larger network, after conflict resolution, identifying which link is 'lan-42.example.com' may be challenging.

3.3.2. Node name (scheme 2)

Our recommendation is to use some short form which indicates the type of node it is, for example, "openwrt.example.com". As the name is visible to users, it should be kept as short as possible. In theory even more exact model could be helpful, for example, "openwrt-buffalo-wzr-600-dhr.example.com". In practice providing some other records indicating exact node information (and access to management UI) is more sensible.

3.3.3. Link name (scheme 2)

Recommendation for (link) portion of (link).(node).(domain) is to use physical network layer type as base, or possibly even just interface name on the node if it's descriptive enough. For example, "eth0.openwrt.example.com" and "wlan0.openwrt.example.com" may be good enough.

4. TLVs

To implement this specification fully, support for following three different TLVs is needed. However, only the DNS Delegated Zone TLVs MUST be supported, and the other two SHOULD be supported.

4.1. DNS Delegated Zone TLV

This TLV is effectively a combined NS and A/AAAA record for a zone. It MUST be supported by implementations conforming to this specification. Implementations SHOULD provide forward zone per link (or optimizing a bit, zone per link with Multicast DNS traffic). Implementations MAY provide reverse zone per prefix using this same mechanism. If multiple nodes advertise same reverse zone, it should be assumed that they all have access to the link with that prefix. However, as noted in Section 5.3, mainly only the node with highest precedence on the link should publish this TLV.

Contents:

- o Address field is IPv6 address (e.g. 2001:db8::3) or IPv4 address mapped to IPv6 address (e.g. ::FFFF:192.0.2.1) where the authoritative DNS server for Zone can be found. If the address field is all zeros, the Zone is under global DNS hierarchy and can be found using normal recursive name lookup starting at the authoritative root servers (This is mostly relevant with the S bit below).
- o S-bit indicates that this delegated zone consists of a full DNS-SD domain, which should be used as base for DNS-SD domain enumeration (that is, (field)._dns-sd._udp.(zone) exists). Forward zones MAY have this set. Reverse zones MUST NOT have this set. This can be used to provision DNS search path to hosts for non-local services (such as those provided by ISP, or other manually configured service providers).
- o B-bit indicates that this delegated zone should be included in network's DNS-SD browse list of domains at b._dns-sd._udp.(domain). Local forward zones SHOULD have this set. Reverse zones SHOULD NOT have this set.
- o L-bit indicates that this delegated zone should be included in the network's DNS-SD legacy browse list of domains at lb._dns-sd._udp.(DOMAIN-NAME). Local forward zones SHOULD have this bit set, reverse zones SHOULD NOT.
- o Zone is the label sequence of the zone, encoded according to section 3.1. ("Name space definitions") of [RFC1035]. Note that name compression is not required here (and would not have any point in any case), as we encode the zones one by one. The zone MUST end with an empty label.

In case of a conflict (same zone being advertised by multiple parties with different address or bits), conflict should be addressed according to Section 3.1.

4.2. Domain Name TLV

This TLV is used to indicate the base (domain) to be used for the network. If multiple nodes advertise different ones, the conflict resolution rules in Section 3.1 should result in only the one with highest precedence advertising one, eventually. In case of such conflict, user SHOULD be notified somehow about this, if possible, using the configuration interface or some other notification mechanism for the nodes. Like the Zone field in Section 4.1, the Domain Name TLV's contents consist of a single DNS label sequence.

This TLV SHOULD be supported if at all possible. It may be derived using some future DHCPv6 option, or be set by manual configuration. Even on nodes without manual configuration options, being able to read the domain name provided by a different node could make the user experience better due to consistent naming of zones across the network.

By default, if no node advertises domain name TLV, hard-coded default (TBD) should be used.

4.3. Node Name TLV

This TLV is used to advertise a node's name. After the conflict resolution procedure described in Section 3.1 finishes, there should be exactly zero to one nodes publishing each node name. The contents of the TLV should be a single DNS label.

This TLV SHOULD be supported if at all possible. If not supported, and another node chooses to use the (link).(node) naming scheme with this node's name, the contents of the network's domain may look misleading (but due to conflict resolution of per-link zones, still functional).

If the node name has been configured manually, and there is a conflict, user SHOULD be notified somehow about this, if possible, using the configuration interface or some other notification mechanism for the nodes.

5. Desirable behavior

5.1. DNS search path in DHCP requests

The nodes following this specification SHOULD provide the used (domain) as one item in the search path to it's hosts, so that DNS-SD browsing will work correctly. They also SHOULD include any DNS Delegated Zone TLVs' zones, that have S bit set.

5.2. Hybrid proxy

The hybrid proxy implementation SHOULD support both forward zones, and IPv4 and IPv6 reverse zones. It SHOULD also detect whether or not there are any Multicast DNS entities on a link, and make that information available to the network zeroconf daemon (if implemented separately). This can be done by (for example) passively monitoring traffic on all covered links, and doing infrequent service enumerations on links that seem to be up, but without any Multicast DNS traffic (if so desired).

Hybrid proxy nodes MAY also publish it's own name via Multicast DNS (both forward A/AAAA records, as well as reverse PTR records) to facilitate applications that trace network topology.

5.3. Hybrid proxy network zeroconf daemon

The daemon should avoid publishing TLVs about links that have no Multicast DNS traffic to keep the DNS-SD browse domain list as concise as possible. It also SHOULD NOT publish delegated zones for links for which zones already exist by another node with higher precedence.

The daemon (or other entity with access to the TLVs) SHOULD generate zone information for DNS implementation that will be used to serve the (domain) zone to hosts. Domain Name TLV described in Section 4.2 should be used as base for the zone, and then all DNS Delegated Zones described in Section 4.1 should be used to produce the rest of the entries in zone (see Appendix A.4 for example interpretation of the TLVs in Appendix A.3).

6. Limited zone stitching for host name resolution

Section 4.1 of the hybrid proxy specification [I-D.ietf-dnssd-hybrid] notes that the stitching of multiple .local zones into a single DNS-SD zone is to be defined later. This specification does not even attempt that, but for the purpose of host name resolution, it is possible to use the set of DNS Delegated Zone TLVs with S-bit or B-bit set to also provide host naming for the (domain). It is done by simply rewriting A/AAAA queries for (name).(domain) to every (name).(ddz-subdomain).(domain), and providing response to the host

when the first non-empty one is received, rewritten back to (name).(domain).

While this scheme is not very scalable, as it multiplies the number of queries by the number of links (given no response in cache), it does work in small networks with relatively few sub-domains.

7. Security Considerations

There is a trade-off between security and zero-configuration in general; if used network state synchronization protocol is not authenticated (and in zero-configuration case, it most likely is not), it is vulnerable to local spoofing attacks. We assume that this scheme is used either within (lower layer) secured networks, or with not-quite-zero-configuration initial set-up.

If some sort of dynamic inclusion of links to be covered using border discovery or such is used, then effectively service discovery will share fate with border discovery (and also security issues if any).

8. IANA Considerations

This document has no actions for IANA.

9. References

9.1. Normative references

- [I-D.ietf-dnssd-hybrid] Cheshire, S., "Hybrid Unicast/Multicast DNS-Based Service Discovery", draft-ietf-dnssd-hybrid-00 (work in progress), November 2014.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<http://www.rfc-editor.org/info/rfc6762>>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<http://www.rfc-editor.org/info/rfc6763>>.

9.2. Informative references

[I-D.ietf-homenet-hnmp]
Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", draft-ietf-homenet-hnmp-09 (work in progress), August 2015.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <<http://www.rfc-editor.org/info/rfc3633>>.

[RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, DOI 10.17487/RFC3646, December 2003, <<http://www.rfc-editor.org/info/rfc3646>>.

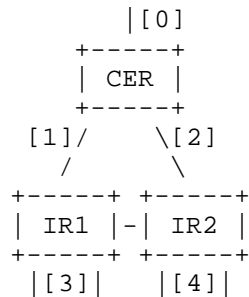
9.3. URIs

[1] <https://github.com/sbyx/hnetd/>

Appendix A. Example configuration

A.1. Used topology

Let's assume home network that looks like this:



We're not really interested about links [0], [1] and [2], or the links between IRs. Given the optimization described in Section 4.1, they should not produce anything to network's Multicast DNS state (and therefore to DNS either) as there isn't any Multicast DNS traffic there.

The user-visible set of links are [3] and [4]; each consisting of a LAN and WLAN link. We assume that ISP provides 2001:db8:1234::/48 prefix to be delegated in the home via [0].

A.2. Zero-configuration steps

Given implementation that chooses to use the second naming scheme (link).(node).(domain), and no configuration whatsoever, here's what happens (the steps are interleaved in practice but illustrated here in order):

1. Network-level state synchronization protocol runs, nodes get effective precedences. For ease of illustration, CER winds up with 2, IR1 with 3, and IR2 with 1.
2. Prefix delegation takes place. IR1 winds up with 2001:db8:1234:11::/64 for LAN and 2001:db8:1234:12::/64 for WLAN. IR2 winds up with 2001:db8:1234:21::/64 for LAN and 2001:db8:1234:22::/64 for WLAN.
3. IR1 is assumed to be reachable at 2001:db8:1234:11::1 and IR2 at 2001:db8:1234:21::1.
4. Each node wants to be called 'node' due to lack of branding in drafts. They announce that using the node name TLV defined in Section 4.3. They also advertise their local zones, but as that information may change, it's omitted here.
5. Conflict resolution ensues. As IR1 has precedence over the rest, it becomes "node". CER and IR2 have to rename, and (depending on timing) one of them becomes "node-2" and other one "node-3". Let us assume IR2 is "node-2". During conflict resolution, each node publishes TLVs for it's own set of delegated zones.
6. CER learns ISP-provided domain "isp.example.com" using DHCPv6 domain list option defined in [RFC3646]. The information is passed along as S-bit enabled delegated zone TLV.

A.3. TLV state

Once there is no longer any conflict in the system, we wind up with following TLVs (NN is used as abbreviation for Node Name, and DZ for Delegated Zone TLVs):

```
(from CER)
DZ {s=1,zone="isp.example.com"}

(from IR1)
NN {name="node"}

DZ {address=2001:db8:1234:11::1, b=1,
    zone="lan.node.example.com."}
DZ {address=2001:db8:1234:11::1,
    zone="1.1.0.0.4.3.2.1.8.b.d.0.1.0.0.2.ip6.arpa."}

DZ {address=2001:db8:1234:11::1, b=1,
    zone="wlan.node.example.com."}
DZ {address=2001:db8:1234:11::1,
    zone="2.1.0.0.4.3.2.1.8.b.d.0.1.0.0.2.ip6.arpa."}

(from IR2)
NN {name="node-2"}

DZ {address=2001:db8:1234:21::1, b=1,
    zone="lan.node-2.example.com."}
DZ {address=2001:db8:1234:21::1,
    zone="1.2.0.0.4.3.2.1.8.b.d.0.1.0.0.2.ip6.arpa."}

DZ {address=2001:db8:1234:21::1, b=1,
    zone="wlan.node-2.example.com."}
DZ {address=2001:db8:1234:21::1,
    zone="2.2.0.0.4.3.2.1.8.b.d.0.1.0.0.2.ip6.arpa."}
```

A.4. DNS zone

In the end, we should wind up with following zone for (domain) which is example.com in this case, available at all nodes, just based on dumping the delegated zone TLVs as NS+AAAA records, and optionally domain list browse entry for DNS-SD:

```
b._dns_sd._udp PTR lan.node
b._dns_sd._udp PTR wlan.node

b._dns_sd._udp PTR lan.node-2
b._dns_sd._udp PTR wlan.node-2

node AAAA 2001:db8:1234:11::1
node-2 AAAA 2001:db8:1234:21::1

node NS node
node-2 NS node-2
```

```
1.1.0.0.4.3.2.1.8.b.d.0.1.0.0.2.ip6.arpa. NS node.example.com.
2.1.0.0.4.3.2.1.8.b.d.0.1.0.0.2.ip6.arpa. NS node.example.com.
1.2.0.0.4.3.2.1.8.b.d.0.1.0.0.2.ip6.arpa. NS node-2.example.com.
2.2.0.0.4.3.2.1.8.b.d.0.1.0.0.2.ip6.arpa. NS node-2.example.com.
```

Internally, the node may interpret the TLVs as it chooses to, as long as externally defined behavior follows semantics of what's given in the above.

A.5. Interaction with hosts

So, what do the hosts receive from the nodes? Using e.g. DHCPv6 DNS options defined in [RFC3646], DNS server address should be one (or multiple) that point at DNS server that has the zone information described in Appendix A.4. Domain list provided to hosts should contain both "example.com" (the hybrid-enabled domain), as well as the externally learned domain "isp.example.com".

When hosts start using DNS-SD, they should check both b._dns-sd._udp.example.com, as well as b._dns-sd._udp.isp.example.com for list of concrete domains to browse, and as a result services from two different domains will seem to be available.

Appendix B. Implementation

There is an prototype implementation of this draft at [hnetd github repository](#) [1] which contains variety of other homenet WG-related things' implementation too.

Appendix C. Why not just proxy Multicast DNS?

Over the time number of people have asked me about how, why, and if we should proxy (originally) link-local Multicast DNS over multiple links.

At some point I meant to write a draft about this, but I think I'm too lazy; so some notes left here for general amusement of people (and to be removed if this ever moves beyond discussion piece).

C.1. General problems

There are two main reasons why Multicast DNS is not proxyable in the general case.

First reason is the conflict resolution depends on the RRsets staying constant. That is not possible across multiple links (due to e.g. link-local addresses having to be filtered). Therefore, conflict resolution breaks, or at least requires ugly hacks to work around.

A simple, but not really working workaround for this is to make sure that in conflict resolution, propagated resources always loses. Given that the proxy function only removes records, the result SHOULD be consistently original set of records winning. Even with that, the conflict resolution will effectively cease working, allowing for two instances of same name to exist (as both think they 'own' the name due to locally seen higher precedence).

Given some more extra logic, it is possible to make this work by having proxies be aware of both the original record sets, and effectively enforcing the correct conflict resolution results by (for example) passing the unfiltered packets to the losing party just to make sure they renumber, or by altering the RR sets so that they will consistently win (by inserting some lower rrclass/rrtype records). As the conflicts happen only in rrclass=1/rrtype=28, it is easy enough to add e.g. extra TXT record (rrtype 16) to force precedence even when removing the later rrtype 28 record. Obviously, this new RRset must never wind up near the host with the higher precedence, or it will cause spurious renaming loops.

Second reason is timing, which is relatively tight in the conflict resolution phase, especially given lossy and/or high latency networks.

C.2. Stateless proxying problems

In general, typical stateless proxy has to involve flooding, as Multicast DNS assumes that most messages are received by every host. And it won't scale very well, as a result.

The conflict resolution is also harder without state. It may result in Multicast DNS responder being in constant probe-announce loop, when it receives altered records, notes that it's the one that should own the record. Given stateful proxying, this would be just a

transient problem but designing stateless proxy that won't cause this is non-trivial exercise.

C.3. Stateful proxying problems

One option is to write proxy that learns state from one link, and propagates it in some way to other links in the network.

A big problem with this case lies in the fact that due to conflict resolution concerns above, it is easy to accidentally send packets that will (possibly due to host mobility) wind up at the originator of the service, who will then perform renaming. That can be alleviated, though, given clever hacks with conflict resolution order.

The stateful proxying may be also too slow to occur within the timeframe allocated for announcing, leading to excessive later renamings based on delayed finding of duplicate services with same name

A work-around exists for this though; if the game doesn't work for you, don't play it. One option would be simply not to propagate ANY records for which conflict has seen even once. This would work, but result in rather fragile, lossy service discovery infrastructure.

There are some other small nits too; for example, Passive Observation Of Failure (POOF) will not work given stateful proxying. Therefore, it leads to requiring somewhat shorter TTLs, perhaps.

Appendix D. Acknowledgements

Thanks to Stuart Cheshire for the original hybrid proxy draft and interesting discussion in Orlando, where I was finally convinced that stateful Multicast DNS proxying is a bad idea.

Also thanks to Mark Baugher, Ole Troan, Shwetha Bhandari and Gert Doering for review comments.

Appendix E. Changelog [RFC Editor: please remove]

draft-ietf-homenet-hybrid-proxy-zeroconf-02:

- o Added subsection on simple zone stitching for host naming purposes.

draft-ietf-homenet-hybrid-proxy-zeroconf-01:

- o Refreshed the draft while waiting on progress of draft-ietf-dnssd-hybrid.

Author's Address

Markus Stenberg
Independent
Helsinki 00930
Finland

Email: markus.stenberg@iki.fi

HOMENET
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

D. Migault (Ed)
Ericsson
T. Mrugalski
ISC
C. Griffiths

R. Weber
Nominum
W. Cloetens
SoftAtHome
October 19, 2015

DHCPv6 Options for Homenet Naming Architecture
draft-ietf-homenet-naming-architecture-dhc-options-03.txt

Abstract

Customer Premises Equipment (CPE) devices are usually constrained devices with reduced network and CPU capabilities. As such, a CPE exposing the authoritative naming service for its home network on the Internet may become vulnerable to resource exhaustion attacks. One way to avoid exposing CPE is to outsource the authoritative service to a third party, e.g. ISP. Such an outsource requires setting up an architecture which may be inappropriate for most end users. This document defines DHCPv6 options so any agnostic CPE can automatically proceed to the appropriate configuration and outsource the authoritative naming service for the home network. In most cases, the outsourcing mechanism is transparent for the end user.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 21, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements notation	3
2. Terminology	3
3. Introduction	3
4. Protocol Overview	6
4.1. Architecture and DHCPv6 Options Overview	6
4.2. Mechanisms Securing DNS Transactions	9
4.3. Primary / Secondary Synchronization versus DNS Update . .	10
5. CPE Configuration	11
5.1. CPE Primary / Secondary Synchronization Configurations .	11
5.1.1. CPE / Synchronization Server	11
5.1.2. CPE / Reverse Synchronization Server	11
5.2. CPE DNS Data Handling and Update Policies	12
5.2.1. Homenet Zone Template	12
5.2.2. DNS (Reverse) Homenet Zone	12
6. Payload Description	13
6.1. Supported Authentication Methods Field	13
6.2. Update Field	14
6.3. Client Public Key Option	14
6.4. Zone Template Option	14
6.5. Synchronization Server Option	15
6.6. Reverse Synchronization Server Option	16
7. DHCP Behavior	17
7.1. DHCPv6 Server Behavior	17
7.2. DHCPv6 Client Behavior	18
7.3. DHCPv6 Relay Agent Behavior	18
8. IANA Considerations	18
9. Security Considerations	19
9.1. DNSSEC is recommended to authenticate DNS hosted data .	19
9.2. Channel between the CPE and ISP DHCP Server MUST be secured	19
9.3. CPEs are sensitive to DoS	19

10. Acknowledgments	19
11. References	19
11.1. Normative References	20
11.2. Informational References	21
Appendix A. Scenarios and impact on the End User	22
A.1. Base Scenario	22
A.2. Third Party Registered Homenet Domain	23
A.3. Third Party DNS Infrastructure	23
A.4. Multiple ISPs	25
Appendix B. Document Change Log	26
Authors' Addresses	27

1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Terminology

The reader is expected to be familiar with [I-D.ietf-homenet-front-end-naming-delegation] and its terminology section. This section defines terms that have not been defined in [I-D.ietf-homenet-front-end-naming-delegation]:

- Client Public Key: designates a public key generated by the CPE. This key is used as an authentication credential for the CPE.
- Homenet Zone Template: The template used as a basis to generate the Homenet Zone.
- DNS Template Server: The DNS server that hosts the Homenet Zone Template.
- Homenet Reverse Zone: The reverse zone file associated to the Homenet Zone.

3. Introduction

CPEs are usually constrained devices with reduced network and CPU capacities. As such, a CPE hosting on the Internet the authoritative naming service for its home network may become vulnerable to resource exhaustion attacks. Outsourcing the authoritative service to a third party avoids exposing the CPE to such attacks. This third party can be the ISP or any other independent third party.

Outsourcing the authoritative naming service to a third party requires setting up an architecture designated in this document as

the Outsourcing Infrastructure. These settings may be inappropriate for most end users that do not have the sufficient knowledge. To address this issue, this document proposes DHCPv6 options so any agnostic CPE can automatically set the Outsourcing Infrastructure. In most cases, these DHCPv6 options are sufficient and do not require any additional interaction from the end user, thus achieving a zero-config settings. In some other cases, the end user is expected to perform some limited manual configuration.

When the CPE is plugged, the DHCPv6 options described in the document enable:

- 1. To build the Homenet Zone: Building the Homenet Zone requires filling the zone with appropriated bindings such as bindings between the names and the IP addresses of the different devices of the home networks. How the CPE is aware of these binding is out of scope of the document. They may be provided, for example, by the DHCPv6 server hosted on the CPE. On the other hand, building the Homenet Zone also requires configuration parameters like the name of the Registered Domain Name associated to the home network or the Public Authoritative Server(s) the Homenet Zone is outsourced to. These configuration parameters are stored in the Homenet Zone Template. This document describes the Zone Template Option which carries the FQDN associated to the Homenet Zone Template. In order to retrieve the Homenet Zone Template, the CPE sends a query of type AXFR [RFC1034], [RFC5936].
- 2. To upload the Homenet Zone to the Synchronization Server, in charge of publishing the Homenet Zone on the Public Authoritative Server(s). This document describes the Synchronization Server Option that provides the FQDN of the appropriated server. Note that, the document does not consider whether the Homenet Zone is signed or not, and if signed, which entity is responsible to sign it. Such questions are out of the scope of the current document.
- 3. To upload the Homenet Reverse Zone to the Reverse Synchronization Server in charge of publishing the Homenet Reverse Zone on the Reverse Public Authoritative Server(s). This document describes the Reverse Synchronization Server Option that provides the FQDN of the appropriated server. Similarly to item 2., we do not consider in this document if the Homenet Reverse Zone is signed or not, and if signed who signs it.
- 4. To provide authentication credential (a public key) to the DHCP Server: Information stored in the Homenet Zone Template, the

Homenet Zone and Homenet Reverse Zone belongs to the CPE, and only the CPE should be able to update or upload these zones. To authenticate the CPE, this document defines the Client Public Key Option. This option is sent by the CPE to the DHCPv6 server and provides the Client Public Key the CPE uses to authenticate itself. This document does not describe mechanisms used to transmit the Client Public Key from the DHCPv6 server to the appropriate entities. If the DHCPv6 server is not able to provide the Client Public Key to the appropriated entities, then the end user is likely to provide manually the Client Public Key to these entities. This document illustrates two scenarios: one where the DHCPv6 server is responsible for distributing the Client Public Key to the Synchronization Servers and Reverse Synchronization Server. In the other scenarios, the Client Public Key is distributed out of band.

The DHCPv6 options described in this document make possible to configure an Outsourcing Infrastructure with no or little configurations from the end user. A zero-config setting is achieved if the the link between the CPE and the DHCPv6 server and the link between the DHCPv6 server and the various DNS servers (Homenet Zone Server, the Reverse Synchronization Server, Synchronization Server) are trusted. For example, one way to provide a trustworthy connection between the CPE and the DHCPv6 server is defined in [I-D.ietf-dhc-sedhcpv6]. When both links are trusted, the CPE is able to provide its authentication credentials (a Client Public Key) to the DHCPv6 server, that in turn forwards it to the various DNS servers. With the authentication credentials on the DNS servers, the CPE is able to securely update.

If the DHCPv6 server cannot provide the Client Public Key to one of these servers (most likely the Synchronization Server) and the CPE needs to interact with the server, then, the end user is expected to provide the CPE's Client Public Key to these servers (the Reverse Synchronization Server or the Synchronization Server) either manually or using other mechanisms. Such mechanisms are outside the scope of this document. In that case, the authentication credentials need to be provided every time the key is modified. Appendix A provides more details on how different scenarios impact the end users.

The remaining of this document is structured as follows. Section 4 provides an overview of the DHCPv6 options as well as the expected interactions between the CPE and the various involved entities. This section also provides an overview of available mechanisms to secure DNS transactions and update DNS data. Section 5 describes how the CPE may securely synchronize and update DNS data. Section 6 describes the payload of the DHCPv6 options and Section 7 details how

DHCPv6 client, server and relay agent behave. Section 8 lists the new parameters to be registered at the IANA, Section 9 provides security considerations. Finally, Appendix A describes how the CPE may behave and be configured regarding various scenarios.

4. Protocol Overview

This section provides an overview of the CPE's interactions with the Outsourcing Infrastructure in Section 4.1, and so the necessary for its setting. In this document, the configuration is provided via DHCPv6 options. Once configured, the CPE is expected to be able to update and publish DNS data on the different components of the Outsourcing Infrastructure. As a result authenticating and updating mechanisms play an important role in the specification. Section 4.2 provides an overview of the different authentication methods and Section 4.3 provides an overview of the different update mechanisms considered to update the DNS data.

4.1. Architecture and DHCPv6 Options Overview

This section illustrates how a CPE receives the necessary information via DHCPv6 options to outsource its authoritative naming service on the Outsourcing Infrastructure. For the sake of simplicity, this section assumes that the DHCPv6 server is able to communicate to the various DNS servers and to provide them the public key associated with the CPE. Once each server got the public key, the CPE can proceed to transactions in an authenticated and secure way.

This scenario has been chosen as it is believed to be the most popular scenario. This document does not ignore that scenarios where the DHCP Server does not have privileged relations with the Synchronization Server must be considered. These cases are discussed latter in Appendix A. Such scenario does not necessarily require configuration for the end user and can also be zero-config.

The scenario is represented in Figure 1.

- 1: The CPE provides its Client Public Key to the DHCP Server using a Client Public Key Option (OPTION_PUBLIC_KEY) and includes the following option codes in its Option Request Option (ORO): Zone Template Option (OPTION_DNS_ZONE_TEMPLATE), the Synchronization Server Option (OPTION_SYNC_SERVER) and the Reverse Synchronization Server Option (OPTION_REVERSE_SYNC_SERVER).
- 2: The DHCP Server makes the Client Public Key available to the DNS servers, so the CPE can secure its DNS transactions. How the Client Public Key is transmitted to the various DNS servers

is out of scope of this document. Note that the Client Public Key alone is not sufficient to perform the authentication and the key should be, for example, associated with an identifier, or the concerned domain name. How the binding is performed is out of scope of the document. It can be a centralized database or various bindings may be sent to the different servers. Figure 1 represents the specific case where the DHCP Server forwards the set (Client Public Key, Zone Template FQDN) to the DNS Template Server, the set (Client Public Key, IPv6 subnet) to the Reverse Synchronization Server and the set (Client Public Key, Registered Homenet Domain) to the Synchronization Server.

- 3: The DHCP Server responds to the CPE with the requested DHCPv6 options, i.e. the Client Public Key Option (OPTION_PUBLIC_KEY), Zone Template Option (OPTION_DNS_ZONE_TEMPLATE), Synchronization Server Option (OPTION_SYNC_SERVER), Reverse Synchronization Server Option (OPTION_REVERSE_SYNC_SERVER). Note that this step may be performed in parallel to step 2, or even before. In other words, there is no requirements that step 3 is conducted after step 2.
- 4: Upon receiving the Zone Template Option (OPTION_DNS_ZONE_TEMPLATE), the CPE performs an AXFR DNS query for the Zone Template FQDN. The exchange is authenticated according to the authentication methods defined in the Supported Authentication Methods field of the DHCP option. Once the CPE has retrieved the DNS Zone Template, the CPE can build the Homenet Zone and the Homenet Reverse Zone. Eventually the CPE signs these zones.
- 5: Once the Homenet Reverse Zone has been set, the CPE uploads the zone to the Reverse Synchronization Server. The Reverse Synchronization Server Option (OPTION_REVERSE_SYNC_SERVER) provides the Reverse Synchronization Server FQDN as well as the upload method, and the Supported Authentication Methods protocol to secure the upload.
- 6: Once the Homenet Zone has been set, the CPE uploads the zone to the Synchronization Server. The Synchronization Server Option (OPTION_SYNC_SERVER) provides the Synchronization Server FQDN as well as the upload method and the authentication method to secure the upload.

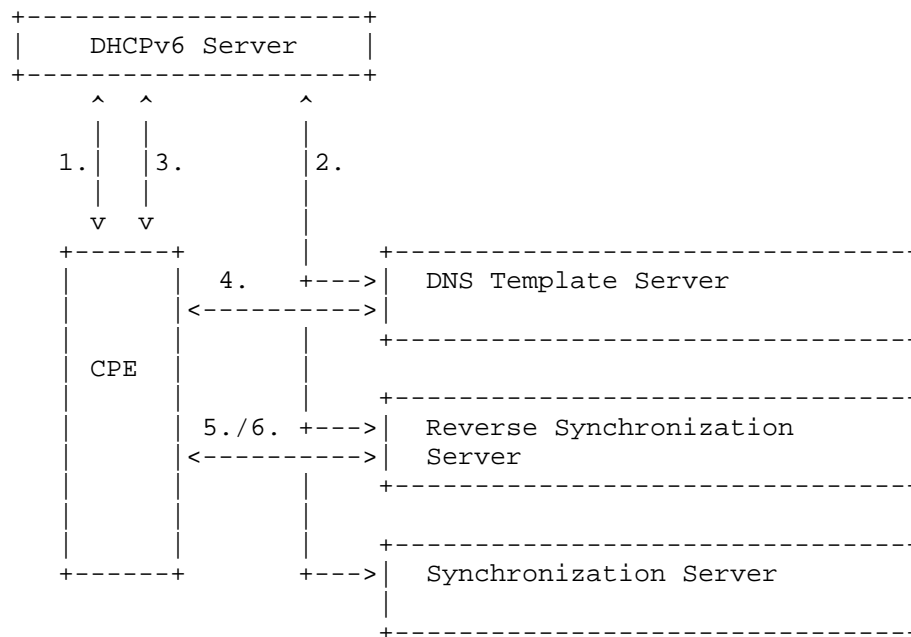


Figure 1: Protocol Overview

As described above, the CPE is likely to interact with various DNS content. More specifically, the CPE is likely to update the:

- Homenet Zone Template: if the configuration of the zone may be changed. This may include additional Public Authoritative Server(s), a different Registered Homenet Domain as the one initially proposed, or a redirection to another domain.
- Homenet Reverse Zone: every time a new device is connected or dis-connected.
- Homenet Zone: every time a new device is connected, dis-connected.

Step 2 and step 3 should be considered as independent steps and could be re-ordered. In fact, the DHCPv6 server does not have to wait for a confirmation from the DNS servers the Client Public Key has been properly received, and is operational by the DNS servers. The DHCP Server is expected to reply upon receiving the Client Public Key Option. The reply to the message with a Client Public Key Option from the DHCP Server is interpreted by the DHCPv6 client as a confirmation of the reception of the option by the DHCP Server only. It does not indicate whether the server had processed the option or

not. Debugging configurations errors or transmission error with one of the DNS servers is let to the CPE and thus is outside of the scope of the DHCPv6. First, it is unlikely a DNS server can validate that the Client Public Key will be operational for the CPE, as multiple causes of errors could occur. For example, the Client Public Key may have been changed during the transmission or by the DHCP Server, or the DNS server may be misconfigured. Second, the number of error codes would be too complex. In addition to multiple causes of errors, multiple architectures and multiple DNS servers may be involved. Third, this may cause significant DHCP Server performance degradation.

In fact, the CPE performs these updates in a secure manner. There are multiple ways to secure a DNS transaction and this document considers two mechanisms: nsupdate and primary/secondary synchronization. Section 4.2 describes the authentication method that may be use to secure the DNS transactions of the CPE. The appropriate authentication methods may, for example, be chosen according to the level of confidentiality or the level of authentication requested by the CPE transactions. Section 4.3 positions the nsupdate and primary/secondary synchronization mechanisms. The update appropriate update mechanism may depend on the for example on the update frequency or the size of the DNS data to update.

4.2. Mechanisms Securing DNS Transactions

Multiple protocols like IPsec [RFC4301] or TLS / DTLS [RFC5246] / [RFC6347] may be used to secure DNS transactions between the CPE and the DNS servers. This document limits its scope to authentication method that have been designed specifically for DNS. This includes DNSSEC [RFC4033], [RFC4034], [RFC4035] that authenticates and provides integrity protection of DNS data, TSIG [RFC2845], [RFC2930] that use a shared secret to secure a transaction between two end points and SIG(0) [RFC2931] authenticates the DNS packet exchanged.

The key issue with TSIG is that a shared secret must be negotiated between the CPE and the server. On the other hand, TSIG performs symmetric cryptography which is light in comparison with asymmetric cryptography used by SIG(0). As a result, over large zone transfer, TSIG may be preferred to SIG(0).

This document does not provide means to distribute shared secret for example using a specific DHCPv6 option. The only assumption made is that the CPE generates or is assigned a public key.

As a result, when the document specifies the transaction is secured with TSIG, it means that either the CPE and the DNS server have been

manually configured with a shared secret, or the shared secret has been negotiated using TKEY [RFC2930], and the TKEY exchanged are secured with SIG(0).

Exchanges with the DNS Template Server to retrieve the Homenet Zone Template may be protected by SIG(0), TSIG or DNSSEC. When DNSSEC is used, it means the DNS Template Server only provides integrity protection, and does not necessarily prevent someone else to query the Homenet Zone Template. In addition, DNSSEC is only a way to protect the AXFR queries transaction, in other words, DNSSEC cannot be used to secure updates. If DNSSEC is used to provide integrity protection for the AXFR response, the CPE should proceed to the DNSSEC signature checks. If signature check fails, it MUST reject the response. If the signature check succeeds, the CPE removes all DNSSEC related RRsets (DNSKEY, RRSIG, NSEC* ...) before building the Homenet Zone. In fact, these DNSSEC related fields are associated to the Homenet Zone Template and not the Homenet Zone.

Any update exchange should use SIG(0) or TSIG to authenticate the exchange.

4.3. Primary / Secondary Synchronization versus DNS Update

As updates only concern DNS zones, this document only considers DNS update mechanisms such as DNS update [RFC2136] [RFC3007] or a primary / secondary synchronization.

The Homenet Zone Template SHOULD be updated with DNS update as it contains static configuration data that is not expected to evolve over time.

The Homenet Reverse Zone and the Homenet Zone can be updated either with DNS update or using a primary / secondary synchronization. As these zones may be large, with frequent updates, we recommend to use the primary / secondary architecture as described in [I-D.ietf-homenet-front-end-naming-delegation]. The primary / secondary mechanism is preferred as it better scales and avoids DoS attacks: First the primary notifies the secondary the zone must be updated, and leaves the secondary to proceed to the update when possible. Then, the NOTIFY message sent by the primary is a small packet that is less likely to load the secondary. At last, the AXFR query performed by the secondary is a small packet sent over TCP (section 4.2 [RFC5936]) which makes unlikely the secondary to perform reflection attacks with a forged NOTIFY. On the other hand, DNS updates can use UDP, packets require more processing than a NOTIFY, and they do not provide the server the opportunity to postpone the update.

5. CPE Configuration

5.1. CPE Primary / Secondary Synchronization Configurations

The primary / secondary architecture is described in [I-D.ietf-homenet-front-end-naming-delegation]. The CPE hosts a Hidden Primary that synchronizes with a Synchronization Server or the Reverse Synchronization Server.

When the CPE is plugged its IP address may be unknown to the secondary. The section details how the CPE or primary communicates the necessary information to set up the secondary.

In order to set the primary / secondary configuration, both primary and secondaries must agree on 1) the zone to be synchronized, 2) the IP address and ports used by both primary and secondary.

5.1.1. CPE / Synchronization Server

The CPE is aware of the zone to be synchronized by reading the Registered Homenet Domain in the Homenet Zone Template provided by the Zone Template Option (OPTION_DNS_ZONE_TEMPLATE). The IP address of the secondary is provided by the Synchronization Server Option (OPTION_SYNC_SERVER).

The Synchronization Server has been configured with the Registered Homenet Domain and the Client Public Key that identifies the CPE. The only missing information is the IP address of the CPE. This IP address is provided by the CPE by sending a NOTIFY [RFC1996].

When the CPE has built its Homenet Zone, it sends a NOTIFY message to the Synchronization Servers. Upon receiving the NOTIFY message, the secondary reads the Registered Homenet Domain and checks the NOTIFY is sent by the authorized primary. This can be done using the shared secret (TSIG) or the public key (SIG(0)). Once the NOTIFY has been authenticated, the Synchronization Servers might consider the source IP address of the NOTIFY query to configure the primaries attributes.

5.1.2. CPE / Reverse Synchronization Server

The CPE is aware of the zone to be synchronized by looking at its assigned prefix. The IP address of the secondary is provided by the Reverse Synchronization Server Option (OPTION_REVERSE_SYNC_SERVER).

Configuration of the secondary is performed as illustrated in Section 5.1.1.

5.2. CPE DNS Data Handling and Update Policies

5.2.1. Homenet Zone Template

The Homenet Zone Template contains at least the related fields of the Public Authoritative Server(s) as well as the Homenet Registered Domain, that is SOA, and NS fields. This template might be generated automatically by the owner of the DHCP Server. For example, an ISP might provide a default Homenet Registered Domain as well as default Public Authoritative Server(s). This default settings should provide the CPE the necessary pieces of information to set the homenet naming architecture.

If the Homenet Zone Template is not subject to modifications or updates, the owner of the template might only use DNSSEC to enable integrity check.

On the other hand, the Homenet Zone Template might also be subject to modification by the CPE. The advantage of using the standard DNS zone format is that standard DNS update mechanism can be used to perform updates. These updates might be accepted or rejected by the owner of the Homenet Zone Template. Policies that defines what is accepted or rejected is out of scope of this document. However, this document assumes the Registered Homenet Domain is used as an index by the Synchronization Server, and SIG(0), TSIG are used to authenticate the CPE. As a result, the Registered Homenet Domain should not be modified unless the Synchronization Server can handle with it.

5.2.2. DNS (Reverse) Homenet Zone

The Homenet Zone might be generated from the Homenet Zone Template. How the Homenet Zone is generated is out of scope of this document. In some cases, the Homenet Zone might be the exact copy of the Homenet Zone Template. In other cases, it might be generated from the Homenet Zone Template with additional RRsets. In some other cases, the Homenet Zone might be generated without considering the Homenet Zone Template, but only considering specific configuration rules.

In the current document the CPE only sets a single zone that is associated with one single Homenet Registered Domain. The domain might be assigned by the owner of the Homenet Zone Template. This constraint does not prevent the CPE to use multiple domain names. How additional domains are considered is out of scope of this document. One way to handle these additional zones is to configure static redirections to the Homenet Zone using CNAME [RFC2181], [RFC1034], DNAME [RFC6672] or CNAME+DNAME [I-D.sury-dnsext-cname-dname].

6. Payload Description

This section details the payload of the DHCPv6 options. A few DHCPv6 options are used to advertise a server the CPE may be expected to interact with. Interaction may require to define update and authentication methods. Update fields are shared by multiple DHCPv6 options and are described in separate sections. Section 6.1 describes the Supported Authentication Method field, Section 6.2 describes the Update field, the remaining Section 6.3, Section 6.4, Section 6.5, Section 6.6 describe the DHCPv6 options.

6.1. Supported Authentication Methods Field

The Supported Authentication Methods field of the DHCPv6 option represented in Figure 2 indicates the authentication method supported by the DNS server. One of these mechanism MUST be chosen by the CPE in order to perform a transaction with the DNS server. See Section 4.2 for more details.

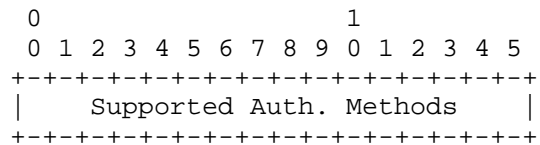


Figure 2: Supported Authentication Methods Filed

- DNS (Bit 0): indicates, when set to 1, that DNS without any security extension is supported.
- DNSSEC (Bit 1): indicates, when set to 1, that DNSSEC provides integrity protection. This can only be used for read operations like retrieving the Homenet Zone Template.
- SIG(0) (Bit 2): indicates, when set to 1, that transaction protected by SIG(0) are supported.
- TSIG (Bit 3): indicates, when set to 1, that transaction using TSIG is supported. Note that if a shared secret has not been previously negotiated between the two party, it should be negotiated using TKEY. The TKEY exchanges MUST be protected with SIG(0) even though SIG(0) is not supported.
- Remaining Bits (Bit 4-15): MUST be set to 0 by the DHCP Server and MUST be ignored by the DHCPv6 client.

A Supported Authentication Methods field with all bits set to zero indicates the operation is not permitted. The Supported

Authentication Methods field may be set to zero when updates operations are not permitted for the DNS Homenet Template. In any other case this is an error.

6.2. Update Field

The Update Field of the DHCPv6 option is represented in Figure 3. It indicates the update mechanism supported by the DNS server. See Section 4.3 for more details.

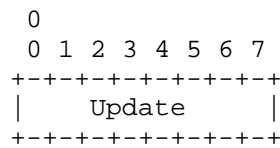


Figure 3: Update Field

- Primary / Secondary (Bit 0): indicates, when set to 1, that DNS Server supports data synchronization using a Primary / Secondary mechanism.
- DNS Update (Bit 1): indicates, when set to 1, that DNS Server supports data synchronization using DNS Updates.
- Remaining Bits (Bit 2-7): MUST be set to 0 by the DHCPv6 server and MUST be ignored by the DHCPv6 client.

6.3. Client Public Key Option

The Client Public Key Option (OPTION_PUBLIC_KEY) indicates the Client Public Key that is used to authenticate the CPE. This option is defined in [I-D.ietf-dhc-sedhcpv6].

6.4. Zone Template Option

The Zone Template Option (OPTION_DNS_ZONE_TEMPLATE) Option indicates the CPE how to retrieve the Homenet Zone Template. It provides a FQDN the CPE SHOULD query with a DNS query of type AXFR as well as the authentication methods associated to the AXFR query or the nsupdate queries. Homenet Zone Template update, if permitted MUST use the DNS Update mechanism.

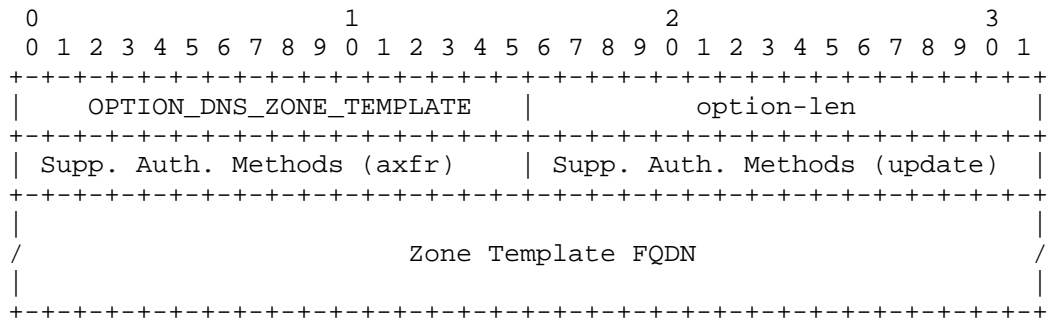


Figure 4: Zone Template Option

- option-code: (16 bits): `OPTION_DNS_ZONE_TEMPLATE`, the option code for the Zone Template Option (TBD1).
- option-len (16 bits): length in octets of the option-data field as described in [RFC3315].
- Supported Authentication Methods(axfr) (16 bits): defines which authentication methods are supported by the DNS server. This field concerns the AXFR and consultation queries, not the update queries. See Section 6.1 for more details.
- Supported Authentication Methods (16 bits): defines which authentication methods are supported by the DNS server. This field concerns the update. See Section 6.1 for more details.
- Zone Template FQDN FQDN (variable): the FQDN of the DNS server hosting the Homenet Zone Template.

6.5. Synchronization Server Option

The Synchronization Server Option (`OPTION_SYNC_SERVER`) provides information necessary for the CPE to upload the Homenet Zone to the Synchronization Server. Finally, the option provides the authentication methods that are available to perform the upload. The upload is performed via a DNS primary / secondary architecture or DNS updates.

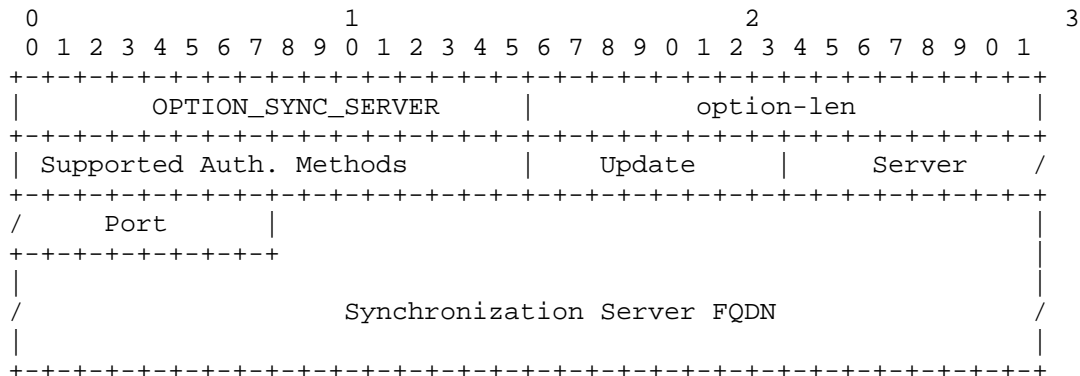


Figure 5: Synchronization Server Option

- option-code (16 bits): `OPTION_SYNC_SERVER`, the option code for the Synchronization Server Option (TBD2).
- option-len (16 bits): length in octets of the option-data field as described in [RFC3315].
- Supported Authentication Methods (16 bits): defines which authentication methods are supported by the DNS server. See Section 6.1 for more details.
- Update (8 bits): defines which update mechanisms are supported by the DNS server. See Section 4.3 for more details.
- Server Port (16 bits): defines the port the Synchronization Server is listening. When multiple transport layers may be used, a single and unique Server Port value applies to all the transport layers. In the case of DNS for example, Server Port value considers DNS exchanges using UDP and TCP.
- Synchronization Server FQDN (variable): the FQDN of the Synchronization Server.

6.6. Reverse Synchronization Server Option

The Reverse Synchronization Server Option (`OPTION_REVERSE_SYNC_SERVER`) provides information necessary for the CPE to upload the Homenet Zone to the Synchronization Server. The option provides the authentication methods that are available to perform the upload. The upload is performed via a DNS primary / secondary architecture or DNS updates.

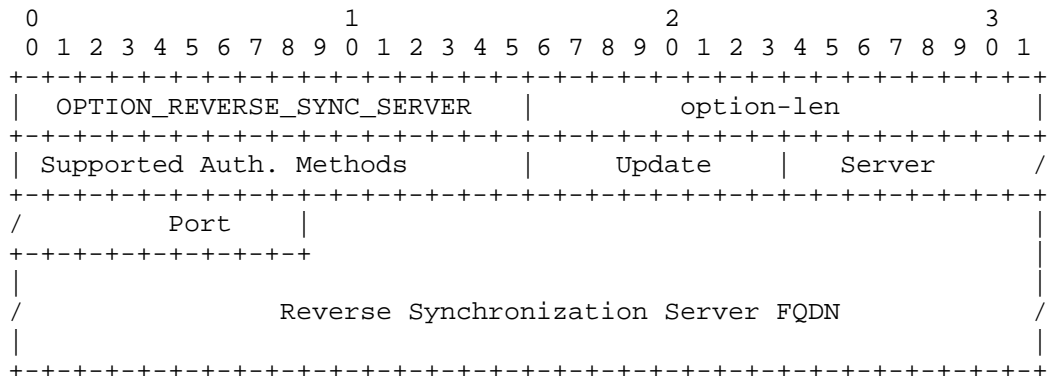


Figure 6: Reverse Synchronization Server Option

- option-code (16 bits): `OPTION_REVERSE_SYNC_SERVER`, the option code for the Reverse Synchronization Server Option (TBD3).
- option-len (16 bits): length in octets of the option-data field as described in [RFC3315].
- Supported Authentication Methods (16 bits): defines which authentication methods are supported by the DNS server. See Section 6.1 for more details.
- Update (8 bits): defines which update mechanisms are supported by the DNS server. See Section 4.3 for more details.
- Server Port (16 bits): defines the port the Synchronization Server is listening.
- Reverse Synchronization Server FQDN (variable): The FQDN of the Reverse Synchronization Server.

7. DHCP Behavior

7.1. DHCPv6 Server Behavior

Sections 17.2.2 and 18.2 of [RFC3315] govern server operation in regards to option assignment. As a convenience to the reader, we mention here that the server will send option foo only if configured with specific values for foo and if the client requested it. In particular, when configured the DHCP Server sends the Zone Template Option, Synchronization Server Option, Reverse Synchronization Server Option when requested by the DHCPv6 client by including necessary option codes in its ORO.

The DHCP Server may receive a Client Public Key Option (OPTION_PUBLIC_KEY) from the CPE. Upon receipt of this DHCPv6 option, the DHCP Server SHOULD acknowledge the reception of the Client Public Key Option as described in Section 4.1 and communicate this credential to the available DNS Servers like the DNS Template Server, the Synchronization Server and the Reverse Synchronization Server, unless not configured to do so.

A CPE may update its Client Public Key by sending a new value in the Client Public Key Option (OPTION_PUBLIC_KEY) as this document assumes the link between the CPE and the DHCP Server is considered authenticated and trusted. The server SHOULD process received Client Public Key Option sent by the client (see step 2 in Section 4.1), unless not configured to do so.

7.2. DHCPv6 Client Behavior

The DHCPv6 client SHOULD send a Client Public Key Option (OPTION_PUBLIC_KEY) to the DHCP Server. This Client Public Key authenticates the CPE.

The DHCPv6 client sends a ORO with the necessary option codes: Zone Template Option, Synchronization Server Option and Reverse Synchronization Server Option.

Upon receiving a DHCP option described in this document in the Reply message, the CPE SHOULD retrieve or update DNS zones using the associated Supported Authentication Methods and update protocols, as described in Section 5.

7.3. DHCPv6 Relay Agent Behavior

There are no additional requirements for the DHCP Relay agents.

8. IANA Considerations

The DHCP options detailed in this document is:

- OPTION_DNS_ZONE_TEMPLATE: TBD1
- OPTION_SYNC_SERVER: TBD2
- OPTION_REVERSE_SYNC_SERVER: TBD3

9. Security Considerations

9.1. DNSSEC is recommended to authenticate DNS hosted data

It is recommended that the (Reverse) Homenet Zone is signed with DNSSEC. The zone may be signed by the CPE or by a third party. We recommend the zone to be signed by the CPE, and that the signed zone is uploaded.

9.2. Channel between the CPE and ISP DHCP Server MUST be secured

The channel MUST be secured because the CPE provides authentication credentials. Unsecured channel may result in CPE impersonation attacks.

The document considers that the channel between the CPE and the ISP DHCP Server is trusted. More specifically, the CPE is authenticated and the exchanged messages are protected. The current document does not specify how to secure the channel. [RFC3315] proposes a DHCP authentication and message exchange protection, [RFC4301], [RFC7296] propose to secure the channel at the IP layer.

9.3. CPEs are sensitive to DoS

CPE have not been designed for handling heavy load. The CPE are exposed on the Internet, and their IP address is publicly published on the Internet via the DNS. This makes the Home Network sensitive to Deny of Service Attacks. The resulting outsourcing architecture is described in [I-D.ietf-homenet-front-end-naming-delegation]. This document shows how the outsourcing architecture can be automatically set.

10. Acknowledgments

We would like to thank Marcin Siodelski and Bernie Volz for their comments on the design of the DHCPv6 options. We would also like to thank Mark Andrews, Andrew Sullivan and Lorenzo Colliti for their remarks on the architecture design. The designed solution has been largely been inspired by Mark Andrews's document [I-D.andrews-dnsop-pd-reverse] as well as discussions with Mark. We also thank Ray Hunter for its reviews, its comments and for suggesting an appropriated terminology.

11. References

11.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <<http://www.rfc-editor.org/info/rfc1996>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<http://www.rfc-editor.org/info/rfc2136>>.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <<http://www.rfc-editor.org/info/rfc2181>>.
- [RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <<http://www.rfc-editor.org/info/rfc2845>>.
- [RFC2930] Eastlake 3rd, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, DOI 10.17487/RFC2930, September 2000, <<http://www.rfc-editor.org/info/rfc2930>>.
- [RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <<http://www.rfc-editor.org/info/rfc2931>>.
- [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <<http://www.rfc-editor.org/info/rfc3007>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<http://www.rfc-editor.org/info/rfc3315>>.

- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<http://www.rfc-editor.org/info/rfc4033>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<http://www.rfc-editor.org/info/rfc4034>>.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<http://www.rfc-editor.org/info/rfc4035>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<http://www.rfc-editor.org/info/rfc4301>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <<http://www.rfc-editor.org/info/rfc5936>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<http://www.rfc-editor.org/info/rfc6347>>.
- [RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, DOI 10.17487/RFC6672, June 2012, <<http://www.rfc-editor.org/info/rfc6672>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<http://www.rfc-editor.org/info/rfc7296>>.

11.2. Informational References

- [I-D.andrews-dnsop-pd-reverse]
Andrews, M., "Automated Delegation of IP6.ARPA reverse zones with Prefix Delegation", draft-andrews-dnsop-pd-reverse-02 (work in progress), November 2013.

[I-D.ietf-dhc-sedhcpv6]

Jiang, S., Shen, S., Zhang, D., and T. Jinmei, "Secure DHCPv6", draft-ietf-dhc-sedhcpv6-08 (work in progress), June 2015.

[I-D.ietf-homenet-front-end-naming-delegation]

Migault, D., Weber, R., Hunter, R., Griffiths, C., and W. Cloetens, "Outsourcing Home Network Authoritative Naming Service", draft-ietf-homenet-front-end-naming-delegation-04 (work in progress), September 2015.

[I-D.sury-dnsextns-cname-dname]

Sury, O., "CNAME+DNS Name Redirection", draft-sury-dnsextns-cname-dname-00 (work in progress), April 2010.

Appendix A. Scenarios and impact on the End User

This section details various scenarios and discuss their impact on the end user.

A.1. Base Scenario

The base scenario is the one described in Section 4. It is typically the one of an ISP that manages the DHCP Server, and all DNS servers.

The end user subscribes to the ISP (foo), and at subscription time registers for example.foo as its Registered Homenet Domain example.foo. Since the ISP knows the Registered Homenet Domain and the Public Authoritative Server(s) the ISP is able to build the Homenet Zone Template.

The ISP manages the DNS Template Server, so it is able to load the Homenet Zone Template on the DNS Template Server.

When the CPE is plugged (at least the first time), it provides its Client Public Key to the DHCP Server. In this scenario, the DHCP Server and the DNS Servers are managed by the ISP so the DHCP Server can provide authentication credentials of the CPE to enable secure authenticated transaction between the CPE and these DNS servers. More specifically, credentials are provided to:

- Synchronization Server
- Reverse Synchronization Server
- DNS Template Server

The CPE can update the zone using DNS update or a primary / secondary configuration in a secure way.

The main advantage of this scenario is that the naming architecture is configured automatically and transparently for the end user.

The drawbacks are that the end user uses a Registered Homenet Domain managed by the ISP and that it relies on the ISP naming infrastructure.

A.2. Third Party Registered Homenet Domain

This section considers the case when the end user wants its home network to use example.com as a Registered Homenet Domain instead of example.foo that has been assigned by the ISP. We also suppose that example.com is not managed by the ISP.

This can also be achieved without any configuration. When the end user buys the domain name example.com, it may request to redirect the name example.com to example.foo using static redirection with CNAME [RFC2181], [RFC1034], DNAME [RFC6672] or CNAME+DNAME [I-D.sury-dnsex-cname-dname].

This configuration is performed once when the domain name example.com is registered. The only information the end user needs to know is the domain name assigned by the ISP. Once this configuration is done no additional configuration is needed anymore. More specifically, the CPE may be changed, the zone can be updated as in Appendix A.1 without any additional configuration from the end user.

The main advantage of this scenario is that the end user benefits from the Zero Configuration of the Base Scenario Appendix A.1. Then, the end user is able to register for its home network an unlimited number of domain names provided by an unlimited number of different third party providers.

The drawback of this scenario may be that the end user still rely on the ISP naming infrastructure. Note that the only case this may be inconvenient is when the DNS Servers provided by the ISPs results in high latency.

A.3. Third Party DNS Infrastructure

This scenario considers that the end user uses example.com as a Registered Homenet Domain, and does not want to rely on the authoritative servers provided by the ISP.

In this section we limit the outsourcing to the Synchronization Server and Public Authoritative Server(s) to a third party. All other DNS Servers DNS Template Server, Reverse Public Authoritative Server(s) and Reverse Synchronization Server remain managed by the ISP. The reason we consider that Reverse Public Authoritative Server(s) and Reverse Synchronization Server remains managed by the ISP are that the prefix is managed by the ISP, so outsourcing these resources requires some redirection agreement with the ISP. More specifically the ISP will need to configure the redirection on one of its Reverse DNS Servers. That said, outsourcing these resources is similar as outsourcing Synchronization Server and Public Authoritative Server(s) to a third party. Similarly, the DNS Template Server can be easily outsourced as detailed in this section

Outsourcing Synchronization Server and Public Authoritative Server(s) requires:

- 1) Updating the Homenet Zone Template: this can be easily done as detailed in Section 4.3 as the DNS Template Server is still managed by the ISP. Such modification can be performed once by any CPE. Once this modification has been performed, the CPE can be changed, the Client Public Key of the CPE may be changed, this does not need to be done another time. One can imagine a GUI on the CPE asking the end user to fill the field with Registered Homenet Domain, optionally Public Authoritative Server(s), with a button "Configure Homenet Zone Template".
- 2) Updating the DHCP Server Information. In fact the Reverse Synchronization Server returned by the ISP is modified. One can imagine a GUI interface that enables the end user to modify its profile parameters. Again, this configuration update is done once-for-ever.
- 3) Upload the authentication credential of the CPE, that is the Client Public Key of the CPE, to the third party. Unless we use specific mechanisms, like communication between the DHCP Server and the third party, or a specific token that is plugged into the CPE, this operation is likely to be performed every time the CPE is changed, and every time the Client Public Key generated by the CPE is changed.

The main advantage of this scenario is that the DNS infrastructure is completely outsourced to the third party. Most likely the Client Public Key that authenticate the CPE need to be configured for every CPE. Configuration is expected to be CPE live-long.

A.4. Multiple ISPs

This scenario considers a CPE connected to multiple ISPs.

Firstly, suppose the CPE has been configured with the based scenarios exposed in Appendix A.1. The CPE has multiple interfaces, one for each ISP, and each of these interface is configured using DHCP. The CPE sends to each ISP its Client Public Key Option as well as a request for a Zone Template Option, a Synchronization Server Option and a Reverse Synchronization Server Option. Each ISP provides the requested DHCP options, with different values. Note that this scenario assumes, the home network has a different Registered Homenet Domain for each ISP as it is managed by the ISP. On the other hand, the CPE Client Public Key may be shared between the CPE and the multiple ISPs. The CPE builds the associate DNS(SEC) Homenet Zone, and proceeds to the various settings as described in Appendix A.1.

The protocol and DHCPv6 options described in this document are fully compatible with a CPE connected to multiple ISPs with multiple Registered Homenet Domains. However, the CPE should be able to handle different Registered Homenet Domains. This is an implementation issue which is outside the scope of the current document. More specifically, multiple Registered Homenet Domains leads to multiple DNS(SEC) Homenet Zones. A basic implementation may erase the DNS(SEC) Homenet Zone that exists when it receives DHCPv6 options, and rebuild everything from scratch. This will work for an initial configuration but comes with a few drawbacks. First, updates to the DNS(SEC) Homenet Zone may only push to one of the multiple Registered Homenet Domain, the latest Registered Homenet Domain that has been set, and this is most likely expected to be almost randomly chosen as it may depend on the latency on each ISP network at the boot time. As a results, this leads to unsynchronized Registered Homenet Domains. Secondly, if the CPE handles in some ways resolution, only the latest Registered Homenet Domain set may be able to provide naming resolution in case of network disruption.

Secondly, suppose the CPE is connected to multiple ISP with a single Registered Homenet Domain. In this case, the one party is chosen to host the Registered Homenet Domain. This entity may be one of the ISP or a third party. Note that having multiple ISPs can be motivated for bandwidth aggregation, or connectivity fail-over. In the case of connectivity fail-over, the fail-over concerns the access network and a failure of the access network may not impact the core network where the Synchronization Server and Public Authoritative Primaries are hosted. In that sense, choosing one of the ISP even in a scenario of multiple ISPs may make sense. However, for sake of simplicity, this scenario assumes that a third party has be chosen to host the Registered Homenet Domain. The DNS settings for each ISP is

described in Appendix A.2 and Appendix A.3. With the configuration described in Appendix A.2, the CPE is expect to be able to handle multiple Homenet Registered Domain, as the third party redirect to one of the ISPs Servers. With the configuration described in Appendix A.3, DNS zone are hosted and maintained by the third party. A single DNS(SEC) Homenet Zone is built and maintained by the CPE. This latter configuration is likely to match most CPE implementations.

The protocol and DHCPv6 options described in this document are fully compatible with a CPE connected to multiple ISPs. To configure or not and how to configure the CPE depends on the CPE facilities. Appendix A.1 and Appendix A.2 require the CPE to handle multiple Registered Homenet Domain, whereas Appendix A.3 does not have such requirement.

Appendix B. Document Change Log

[RFC Editor: This section is to be removed before publication]

-05: changing Master to Primary, Slave to Secondary

-04: Working Version Major modifications are:

- Re-structuring the draft: description and comparison of update and authentication methods have been integrated into the Overview section. a Configuration section has been created to describe both configuration and corresponding behavior of the CPE.
- Adding Ports parameters: Server Set can configure a port. The Port Server parameter have been added in the DHCPv6 option payloads because middle boxes may not be configured to let port 53 packets and it may also be useful to split servers among different ports, assigning each end user a different port.
- Multiple ISP scenario: In order to address comments, the multiple ISPs scenario has been described to explicitly show that the protocol and DHCPv6 options do not prevent a CPE connected to multiple independent ISPs.

-03: Working Version Major modifications are:

- Redesigning options/scope: according to feed backs received from the IETF89 presentation in the dhc WG.
- Redesigning architecture: according to feed backs received from the IETF89 presentation in the homenet WG, discussion with Mark and Lorenzo.

- 02: Working Version Major modifications are:
 - Redesigning options/scope: As suggested by Bernie Volz
- 01: Working Version Major modifications are:
 - Remove the DNS Zone file construction: As suggested by Bernie Volz
 - DHCPv6 Client behavior: Following options guide lines
 - DHCPv6 Server behavior: Following options guide lines
- 00: version published in the homenet WG. Major modifications are:
 - Reformatting of DHCPv6 options: Following options guide lines
 - DHCPv6 Client behavior: Following options guide lines
 - DHCPv6 Server behavior: Following options guide lines
- 00: First version published in dhc WG.

Authors' Addresses

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Email: daniel.migault@ericsson.com

Tomek Mrugalski
Internet Systems Consortium, Inc.
950 Charter Street
Redwood City, CA 94063
USA

Email: tomasz.mrugalski@gmail.com
URI: <http://www.isc.org>

Chris Griffiths

Email: cgriffiths@gmail.com

Ralf Weber
Nominum
2000 Seaport Blvd #400
Redwood City, CA 94063
US

Email: ralf.weber@nominum.com
URI: <http://www.nominum.com>

Wouter Cloetens
SoftAtHome
vaartdijk 3 701
3018 Wijkmaal
Belgium

Email: wouter.cloetens@softathome.com

Homenet Working Group
Internet-Draft
Intended status: Informational
Expires: July 24, 2016

R. Bellis
ISC
W. Townsley
Cisco
January 21, 2016

Homenet Routing Consensus Call
draft-ietf-homenet-routing-consensus-call-01

Abstract

In order to support arbitrary network topologies and multi-homing the IETF Homenet Architecture [RFC7368] requires that a routing protocol operates inside each home network. For interoperability reasons it is necessary for there be a single "mandatory to implement" routing protocol. With the Homenet Working Group unable to reach clear consensus on which protocol that should be the Working Group Chairs (with the support of the Internet Area Director) declared rough consensus that the chosen protocol is BABEL [RFC6126]. This document (not intended for publication as an RFC) serves as an additional record of that decision.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 24, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Statement	2
2. IANA Considerations	3
3. Security Considerations	3
4. Acknowledgements	3
5. Informative References	3
Authors' Addresses	3

1. Statement

On the 27th of October, 2015, the Working Group Chairs and the Internet Area Director made the following statement to the Homenet Mailing List:

The Chairs believe that there is WG consensus that a single "mandatory to implement" routing protocol must be chosen. We also believe that further delaying the direction here has long passed the point of diminishing returns.

Based on the feedback received in Prague and on the WG mailing list thereafter, we are therefore declaring rough consensus that BABEL [RFC6126] shall be the "mandatory to implement" routing protocol for Homenet routers, albeit only on an Experimental basis at this time.

The aim in making this decision is to allow the non-routing-protocol aspects of Homenet to move forward in the near term, while allowing time for additional implementation, experimentation and specification. To that end, we solicit Experimental Internet Drafts to document Homenet-specific profiles of any applicable routing solution and to report results of any relevant experimentation and implementation.

We expect that this decision will be revisited in a future Standards Track document based on specifications and running code available at that time.

Vendors looking to ship Homenet routers in the near term should refer to [RFC6126], [RFC7557], [I-D.boutier-babel-source-specific], and available open source

implementations thereof for the routing protocol portion of the Homenet solution space.

2. IANA Considerations

This document has no IANA considerations.

3. Security Considerations

This document has no security considerations.

4. Acknowledgements

We wish to thank Terry Manderson (INT Area AD) for his support.

5. Informative References

- [RFC7368] Chown, T., Ed., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", RFC 7368, DOI 10.17487/RFC7368, October 2014, <<http://www.rfc-editor.org/info/rfc7368>>.
- [RFC6126] Chroboczek, J., "The Babel Routing Protocol", RFC 6126, DOI 10.17487/RFC6126, April 2011, <<http://www.rfc-editor.org/info/rfc6126>>.
- [RFC7557] Chroboczek, J., "Extension Mechanism for the Babel Routing Protocol", RFC 7557, DOI 10.17487/RFC7557, May 2015, <<http://www.rfc-editor.org/info/rfc7557>>.
- [I-D.boutier-babel-source-specific] Boutier, M. and J. Chroboczek, "Source-Specific Routing in Babel", draft-boutier-babel-source-specific-01 (work in progress), May 2015.

Authors' Addresses

Ray Bellis
Internet Systems Consortium, Inc.
950 Charter Street
Redwood City CA 94063
USA

Phone: +1 640 423 1200
Email: ray@isc.org

Mark Townsley
Cisco

Email: mark@townsley.net

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 22, 2016

T. Lemon
Nominum, Inc.
March 21, 2016

Homenet Naming and Service Discovery Architecture
draft-lemon-homenet-naming-architecture-00

Abstract

This document recommends a naming and service discovery resolution architecture for homenets. This architecture covers the publication and resolution of names of hosts on the homenet both within the homenet and on the public internet, and the use of such names for offering and discovering services that exist on the homenet both within the homenet and on the public internet. Security and privacy implications and techniques for automatically and administratively setting security and privacy policies for such names are also described.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 22, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Existing solutions	4
2. Homenet Naming Database	5
2.1. Global Name	5
2.2. Local namespaces	7
2.3. Public namespaces	7
2.4. Adding Names	8
2.4.1. mDNS Snooping	8
2.4.2. DHCP DNS Update (stateful or stateless)	9
2.4.3. DNS Update	9
2.5. Removing Names	9
2.6. Name Collisions	10
2.7. Recovery from loss	10
2.8. Persistence	10
2.9. Well-known names	11
3. Name Resolution	11
3.1. Configuring Resolvers	11
3.2. Configuring Service Discovery	12
3.3. Resolution of local namespaces	12
3.4. Local and Public Zones	12
3.5. Legacy support	13
3.6. DNSSEC Validation	13
3.7. Support for Multiple Provisioning Domains	14
3.8. Using the Local Namespace While Away From Home	14
4. Publishing the Public Namespace	15
4.1. Acquiring the Global Name	15
4.2. Hidden Primary/Public Secondaries	16
4.3. DNSSEC security	17
4.4. PKI security	17
4.5. Renumbering	18
4.6. ULA	18
5. Management	18
5.1. End-user management	18
5.2. Central management	19
6. Privacy Considerations	19
7. Security Considerations	19
8. IANA considerations	19
9. Normative References	20
Author's Address	20

1. Introduction

Associating domain names with hosts on the Internet is a key factor in enabling communication with hosts, particularly service discovery. In order to provide name service, several provisioning mechanisms must be available:

- o Provisioning of a namespace in which names can be published and services advertised
- o Associating a name within that namespace to the set of IP addresses on which a host is reachable
- o Advertising services available on the local network and associating those services with names published in the namespace
- o Distribution of names published in that namespace to servers that can be queried in order to resolve names
- o Correct advertisement of name servers that can be queried in order to resolve names
- o Timely removal of published names when they are no longer in use

Homenet adds the following considerations:

1. Some names may be published in a broader scope than others. For example, it may be desirable to advertise some homenet services to users who are not connected to the homenet. However, it is unlikely that all services published on the home network would be appropriate to publish outside of the home network. In many cases, no services will be appropriate to publish outside of the network, but the ability to do so is required.
2. Users cannot be assumed to be skilled or knowledgeable in name service operation, or even to have any sort of mental model of how these functions work. With the possible exception of policy decisions, all of the operations mentioned here must reliably function automatically, without any user intervention or debugging. Even to the extent that users may provide input on policy, such as whether a service should or should not be advertised outside of the home, the user must be able to safely provide such input without having a correct mental model of how naming and service discovery work, and without being able to reason about security in a nuanced way.

3. Because user intervention cannot be required, naming conflicts must be resolved automatically, and, to the extent possible, transparently.
4. Where services are advertised both on and off the home network, differences in naming conventions that may vary depending on the user's location must likewise be transparent to the end user.
5. Hosts that do not implement any homenet-specific capabilities must still be able to discover and access services on the homenet, to the extent possible.
6. Homenet explicitly supports multihoming--connecting to more than one Internet Service Provider--and therefore support for multiple provisioning domains [6] is required to deal with situations where the DNS may give a different answer depending on whether caching resolvers at one ISP or another are queried.
7. Multihomed homenets may treat all service provider links as equivalent, or may treat some links as primary and some as backup, either because of differing transit costs or differing performance. Services advertised off-network may therefore be advertised for some links and not others.

In addition to these considerations, there may be a need to provide for secure communication between end users and the user interface of the home network, as well as to provide secure name validation (e.g., DNSSEC). Secure communications require that the entity being secured have a name that is unique and can be cryptographically authenticated within the scope of use of all devices that must communicate with that entity. Because it is very likely that devices connecting to one homenet will be sufficiently portable that they may connect to many homenets, the scope of use must be assumed to be global. Therefore, each homenet must have a globally unique name.

1.1. Existing solutions

Previous attempts to automate naming and service discovery in the context of a home network are able to function with varying degrees of success depending on the topology of the home network. For example, Multicast DNS [4] can provide naming and service discovery [5], but only within a single multicast domain.

The Domain Name System provides a hierarchical namespace [1], a mechanism for querying name servers to resolve names [2], a mechanism for updating namespaces by adding and removing names [3], and a mechanism for discovering services [5]. Unfortunately, DNS provides no mechanism for automatically provisioning new namespaces, and

secure updates to namespaces require pre-shared keys, which won't work for an unmanaged network. DHCP can be used to populate names in a DNS namespace; however at present DHCP cannot provision service discovery information.

Hybrid Multicast DNS [7] proposes a mechanism for solving the single-multicast-domain problem. However, it has serious shortcomings as a solution to the Homenet naming problem. The most obvious shortcoming is that it requires that every multicast domain have a separate name. This then requires that the homenet generate names for every multicast domain, and requires that the end user have a mental model of the topology of the network in order to guess on which link a given service may appear.

2. Homenet Naming Database

In order to resolve names, there must be a place where names are stored. There are two ways to go about this: either names are stored on the devices that own them, or they are stored in the network. This isn't a clean dichotomy, however: it's possible for the source of truth about a name to be owned by the device, while the resolution of the name is owned by a service separate from the device. Additionally, if names are owned by devices, conflicts can arise, since two devices might present the same name by default or by accident. Further, devices can be attached to more than one network, in which case we want the same name to identify them on both networks. Additionally, although homenets are self-configuring, user customization is permitted and useful.

In order to achieve this, the Homenet Naming Database (HNDB) provides a persistent central store into which names can be registered

2.1. Global Name

Every homenet must be able to have a name in the global DNS hierarchy which serves as the root of the zone in which the homenet publishes its public namespaces. Homenets that do not yet have a name in the global namespace use the homenet special-use TLD [TBD1] as their "global name" until they are configured with a global name.

[It's tempting to have the homenet generate a UUID-like name that can be used as a global name, but we really don't want that, because it will be quite ugly to the user, difficult to remember, and therefore not protective against the kinds of mistakes we'd want such a name to protect against. It's better as a security UI to have the user see a name that is the same on all homenets. This will allow the user to fairly easily notice that they see the same name on every homenet. To present a name that isn't really unique and isn't easily

identified as anything other than "random gibberish," may lull the end user into a sense that they are talking to the right homenet when they see the random gibberish, without realizing that actually it's different gibberish and they aren't talking to their own homenet.]

A homenet's global name can be a name that the homenet user has registered on their own in the DNS using a public DNS registrar. However, this is not required and, indeed, presents some operational challenges. It can also be a name under a domain owned by one of the user's service providers, or managed by some service provider that specifically provides homenet naming services.

For most end-users, the second or third options will be preferable. It will allow them to choose an easily-remembered homenet domain name under an easily-remembered service provider subdomain, and will not require them to maintain a DNS registration.

[This doesn't belong in the arch document, at least not in this part, but I'm writing it down because I think we should discuss it and figure out exactly what we should recommend, and I do believe we should recommend something here and not just hope for the best. I am also hoping to stave off frivolous, privacy-harming patents by publishing this now:]

Providers of either of these types of homenet naming service should offer a selection of different provider TLDs rather than a flat namespace, so as to avoid the exampleuser1997 problem. So for example rather than having a single namespace "isp.example.com", the provider should have a series of subdomains, like "grapefruit.example.com", "warrior.example.com", "koala.example.com", "rocket.example.com", and so forth.

Some small number of such subdomains should be presented to the user when registering. Subdomains with more than perhaps 50,000 homenets registered in them should never be presented for registration, to avoid chosen name collisions. If the user is unable to find a namespace they like, it may be beneficial to allow the user to cycle through a larger set of namespaces. The user should wind up with a global name like "hamburger.warrior.example.com".

This specification may seem frivolous or overly-prescriptive. The reason for being this specific is that it is important for the user to be able to quickly choose a memorable name that doesn't contain personal identifying information. Getting this user interface right has significant implications for the user's privacy and security. Any user interface that meets the criterion that the user can quickly choose a memorable name that doesn't contain personal information will address this requirement. The user should be specifically told

not to use personal information like birthdays or names of friends or pets, and should be encouraged to write the chosen name down until they have it memorized.

2.2. Local namespaces

Every homenet has two non-hierarchical local namespaces, one for associating DNS RRs with names, and one for associating DNS RRs with IP addresses. These namespaces are key-data stores, where for the name mapping the primary key is a single DNS label, and for the IP address mapping the primary key is an IP address. The secondary key is an RRtype, so that there can be more than one RRset per IP address, and each data element is an RRset of the corresponding RRtype.

Each RRset in each local namespace is marked with a flag that indicates whether it is to be public or private. Each RRset is also marked with a flag that indicates whether its availability is critical or best-effort. This flag only has meaning if the RRset is marked public.

Each RR that contains a name (e.g, a CNAME or SRV record) either contains a name in the local namespace or a global name. All global names are references to external services, not to services on the homenet. All local names are qualified with the homenet-specific special-use TLD, [TBD1].

[Question: do we need RRset granularity for these flags?]

The local namespace is maintained as a distributed database with copies on every homenet router. No copy is the master copy. Although the local namespace is non-hierarchical, it is permissible for it to contain RRtypes that contain delegations. However, from an operational perspective it is most likely better for the local namespace to be at the bottom of the delegation hierarchy, and so we do not recommend the use of such delegations.

2.3. Public namespaces

Every homenet has two public namespaces. These are copies of the private namespaces with four modifications:

1. Names with no public RRsets are not included in the public namespace.
2. RRs that contain IP addresses in the homenet's ULA prefix are omitted.

3. By default, RRs that contain IPv4 addresses are omitted, because IPv4 doesn't support renumbering. However, there should be a whitelist of IPv4 addresses that may be published, so that if the end user has static IPv4 addresses, those can be published.
4. If an RRset is marked best-effort rather than critical, RRs containing IP addresses that match prefixes assigned by backup links are omitted.

[Are there RRtypes or classes of DNSSD records that we want to always omit?]

Because the public namespaces are copies of the private namespaces, replication is not necessary: each homenet router automatically produces public namespaces by deriving them from the private namespaces using the above rules. The public namespaces can be derived on demand, or maintained automatically as updates are made to the private namespaces.

[Security/privacy considerations: It can be argued that public namespaces provide a means for botnets to publish rendezvous information. However, in fact this isn't really true because if botnets did so, it would be very obvious, so would wind up being used as a means of tracking and suppressing them. It's probably better to encourage this mistake rather than trying to prevent it, since the former benefits white hats, and the latter limits functionality.]

[Presumably we want it to be possible for devices that are meant to be public servers to publish their names in the DNS, but how do we automatically determine that a device is "meant" to be public? This needs thought.]

2.4. Adding Names

Several mechanisms are available for updating the HNDB.

2.4.1. mDNS Snooping

Homenet snoops mDNS for device names. Homenet does not defend names. If two devices with the same name appear on a link, mDNS handles collision resolution. If two devices with the same name appear on different links, homenet deterministically generates names for both devices using the link-layer address of each device, plus the name that device claimed. If a device that appears on two links is the same device (presents the same link-layer address or DHCID) then it is treated as a single device, with a single name. This whole discussion probably belongs in a separate draft.

2.4.2. DHCP DNS Update (stateful or stateless)

A and PTR records can be set up this way. Doesn't work (yet) for service discovery. Should we write a draft, or just rely on:

2.4.3. DNS Update

DNS updates can be send to any resolver in the homenet to add names to the local zone. If there is no conflict, the name is added; otherwise the update is rejected.

[Really, what we ought to do is just allow devices to declare an ID that is a public key, and then do DNS updates to the local service zone signed with the corresponding private key. Devices would also sign their mDNS claims with the same key, so that mDNS updates for devices that support this functionality can be ignored by the mDNS snooping agent.]

[Records added to the namespace that contain names are problematic: the hostname label is obvious, but what about the domain name? I think the answer is that these names shouldn't be qualified, and the name server should qualify them appropriately. What does mDNS do?]

[Add something about [TBD1] versus global name. If there is a global name, that is what we use for name resolution on the local network. This is necessary because of the security implications, both for DNSSEC and for PKI.]

2.5. Removing Names

mDNS: names time out

DHCP: names go away when lease expires, or, for stateless, when refresh timer expires

DNS Update: names have to have a lifetime, determined by the DNS server, and DNSSD devices that do DNS Update have to keep sending updates at appropriate intervals to reset the timer. If the lifetime of a name expires, the name is deleted. Names can also be deleted by a new update, which must either be signed with SIG(0) using a key published on the name, or else must come from an IP address published in the name if no key is published on the name (what if there's no IP address?)

2.6. Name Collisions

Covered under adding names. Say more?

2.7. Recovery from loss

In principle the names in the zone aren't precious. If there are multiple HNRs and one is replaced, the replacement recovers by copying the local namespaces and other info from the others. If all are lost, there are a few pieces of persistent data that need to be recovered:

- o The global name
- o The ZSK for both local namespaces
- o Names configured statically through the UI

All other names were acquired dynamically, and recovery is simply a matter of waiting for the device to re-announce its name. We should have a protocol for informing devices that they should do this, either using an ND option or a DHCP message, or else devices should know to do a fresh announcement whenever the link goes away (but that doesn't work if the device is connected to the nearest router through a separate switch, so ND option is better).

There should be some way (ideally) for the global name to be recovered. How? This will cause problems with DNSSEC, because the private ZSK is lost, and we don't expect the user to be smart about keys. ZSK or KSK could be stored encrypted at the SP. Subject to brute force attacks if so, probably not a good idea. Better to just have a flag day? One answer is the end-user management RESTful API: if the end-user has a phone, the ZSK and other static info can be maintained in an app on the phone; this app can then be in touch with the homenet, and if the homenet finds that it is amnesiac, the app can notify the user. Of course, this is a potential attack--we don't want some other network the phone connects to to be able to steal the ZSK just by telling the app it's amnesiac.

2.8. Persistence

When the whole homenet goes away, can we recover the zone? Step 1: figure out what name we had: how? Then, if we have off-site secondaries that have copies of the private zone, we can poll for the highest serial number and copy the contents of that zone. If we only have secondaries for the public part of the zone, we can recover that; should we?

2.9. Well-known names

Homenets serve a zone under the special-use TLD [TBD2], that answers queries for local configuration information and can be used to advertise services provided by the homenet (as opposed to services present on the homenet). This provides a standard means for querying the homenet that can be assumed by management functions and homenet clients. A registry of well-known names for this zone is defined in IANA considerations (Section 8). Names and RRs in this zone are only ever provided by the homenet--this is not a general purpose service discovery zone.

All resolvers on the homenet will answer questions about names in this zone; entries in the zone are guaranteed not to be globally unique. Hosts and services that use the special names under this TLD are assumed to be aware that it is a special TLD. If such hosts cache DNS entries, DNS entries under this TLD are discarded whenever the host detects a network reattachment.

The `uuid.[TBD2]` name contains a TXT RR that contains the UUID of the homenet. Each homenet generates its own distinct UUID; homenet routers on any particular homenet all use the same UUID, which is agreed upon using HNCP. If the homenet has not yet generated a UUID, queries against this name will return NXDOMAIN.

The `global-name.[TBD2]` name contains a PTR record that contains the global name of the homenet. If the homenet does not have a global name, queries against this name will return NXDOMAIN.

The `global-name-register.[TBD2]` name contains one or more A and/or AAAA records referencing hosts that provide a RESTful API over HTTP that can be used to register the global name of the homenet, once that name has been configured.

3. Name Resolution

3.1. Configuring Resolvers

Hosts on the homenet receive a set of resolver IP addresses using either DHCP or RA. IPv4-only hosts will receive IPv4 addresses of resolvers, if available, over DHCP. IPv6-only hosts will receive resolver IPv6 addresses using either stateful (if available) or stateless DHCPv6, or through the domain name option in router advertisements. All homenet routers provide resolver information using both stateless DHCPv6 and RA; support for stateful DHCPv6 and DHCPv4 is optional (right?).

3.2. Configuring Service Discovery

DNS-SD uses several default domains for advertising local zones that are available for service discovery. These include the ".local" domain, which is searched using mDNS, and also the IPv4 and IPv6 reverse zone corresponding to the prefixes in use on the local network. For the homenet, queries against the ".local" zone are supported, as well as queries against every IPv4 address prefix advertised on a homenet link, and every IPv6 address prefix advertised on a homenet link, including prefixes derived from the homenet's ULA(s). In addition, the [TBD2] domain can be used, and is preferred. Whenever the "<domain>" sequence appears in this section, it references each of the domains mentioned in this paragraph.

Homenets advertise the availability of several browsing zones in the "b._dns_sd.<domain>" subdomain. The zones advertised are the "well known" zone (TBD2) and the zone containing the local namespace. If the global name is available, only that name is advertised for the local namespace; otherwise [TBD1] is advertised. Similarly, if the global name is available, it is advertised as the default browsing and service registration domain under "db._dns_sd.<domain>", "r._dns_sd.<domain>", "dr._dns_sd.<domain>" and "lb._dns_sd.<domain>"; otherwise, the name [TBD1] is advertised as the default.

3.3. Resolution of local namespaces

The local namespace appears in two places, under [TBD1] and, if the homenet has a global name, under the global name. Resolution from inside the homenet yields the contents of the local namespaces; resolution outside of the homenet yields the contents of the public namespaces. If there is a global name for the homenet, RRs containing names in both instances of the local namespace are qualified with the global name; otherwise they are qualified with [TBD1].

3.4. Local and Public Zones

The homenet's global name serves both as a unique identifier for the homenet and as a delegation point in the DNS for the zone containing the homenet's forward namespace. There are two versions of the forward namespace: the public version and the private version. Both of these versions of the namespace appear under the global name delegation, depending on which resolver a host is querying.

The homenet provides two versions of the zone. One is the public version, and one is the local version. The public version is never visible on the homenet (could be an exception for a guest net). The

public version is available outside of the homenet. The local version is visible on the homenet. Whenever the zone is updated, it is signed with the ZSK. Both versions of the zone are signed; the local signed version always has a serial number greater than the public signed version. [we want to not re-sign the public zone if no public names in the private zone changed.]

This dual publication model relies on hosts connected to the homenet using the local resolver and not some external resolver. Hosts that use an external resolver will see the public version of the namespace. From a security UI design perspective, allowing queries from hosts on the homenet to resolvers off the homenet is risky, and should be prevented by default. This is because if the user sees inconsistent behavior on hosts that have external resolvers configured, they may attempt to fix this by making all local names public. If an alternate external resolver is to be used, it should be configured on the homenet, not on the individual host.

One way to make this work is to intercept all DNS queries to non-homenet IP addresses, check to see if they reference the local namespace, and if so resolve them locally, answering as if from the remote cache. If the query does not reference a local namespace, and is listed as "do not forward" in RFC 6761 or elsewhere, it can be sent to the intended cache server for resolution without any special handling for the response. This functionality is not required for homenet routers, but is likely to present a better user experience.

3.5. Legacy support

In principle, devices that support DNSSD should be able to do service discovery using DNS without any special help. Devices that only support mDNS should be able to get a complete list of services from a combination of names published by devices on the same link and by the homenet router that serves that link (what if there's more than one?). In cases where the homenet router has an off-link entry that has the same claimed name as an on-link service, the homenet router does not advertise the off-link service.

3.6. DNSSEC Validation

The [TBD1] zone is not validated. We could define a special rule, such that any particular local zone publishes a unique identifier for that zone and signs itself with a ZSK; a homenet-aware host could do TOFU on the id/ZSK, and could keep a list of id/ZSKs it has seen, and then do DNSSEC validation on names in the local zone that way, but it's a bit rickety and nonstandard, so I don't know if there's enough benefit to justify the cost. Worth thinking about, though--could be the keystone of a homenet security model if done right.

3.7. Support for Multiple Provisioning Domains

Homenets must support the Multiple Provisioning Domain Architecture [6]. In order to support this architecture, each homenet router that provides name resolution must provide one resolver for each provisioning domain (PvD). Each homenet router will advertise one resolver IP address for each PvD. DNS requests to the resolver associated with a particular PvD, e.g. using RA options [8] will be resolved using the external resolver(s) provisioned by the service provider responsible for that PvD.

The homenet is a separate provisioning domain from any of the service providers. The global name of the homenet can be used as a provisioning domain identifier, if one is configured. Homenets should allow the name of the local provisioning domain to be configured; otherwise by default it should be "Home Network xxx", where xxx is the generated portion of the homenet's ULA prefix, represented as a base64 string.

The resolver for the homenet PvD is offered as the primary resolver in RAs and through DHCPv4 and DHCPv6. When queries are made to the homenet-PvD-specific resolver for names that are not local to the homenet, the resolver will use a round-robin technique, alternating between service providers with each step in the round-robin process, and then also between external resolvers at a particular service provider if a service provider provides more than one. The round-robinning should be done in such a way that no service provider is preferred, so if service provider A provides one caching resolver (A), and service provider B provides two (B1, B2), the round robin order will be (A, B1, A, B2), not (A, B1, B2).

Every resolver provided by the homenet, regardless of which provisioning domain it is intended to serve, will accept updates for services in the local service namespace.

3.8. Using the Local Namespace While Away From Home

Homenet routers do not answer unauthenticated DNS queries from off the local network. However, some applications may benefit from the ability to resolve names in the local namespace while off-network. Therefore hosts connected to the homenet can register keys in the same way that services are registered, and the homenet will cache such keys. Such keys must be validated by the end user before queries against the local namespace that have been authenticated with that key are permitted. A host that will make remote queries to the local namespace caches the names of all DNS servers on the homenet by querying all-resolver-names.[TBD2]. If the local zone is not signed using DNSSEC, the host also caches each server's SIG(0) key.

Hosts that require name resolution from the local network must have a stub resolver configured to contact the dns server on one or more routers in the homenet when resolving names in the local namespace. To do this, resolvers must know the global name of the local namespace, which they can retain from previous connections to the homenet. The homenet may not have a stable IP address, so such resolvers cannot merely cache the IP address of the homenet routers. Instead, they cache the names of the homenet routers that provide DNS service and use those names to determine the IP addresses of the homenet routers at the time of resolution. Such IP addresses can be safely cached for the duration of the TTL of the A or AAAA record that contained them. The names of the homenet router DNS servers should be randomly generated so that they can't be guessed by off-network attackers. [?]

To make a homenet DNS query, the host signs the request using SIG(0) with the key that they registered to the homenet. The homenet router first checks the question in the query for validity: it must be a subdomain of the global name. The homenet router then checks the name of the signing key against the list of cached, validated keys; if that key is cached and validated, then the homenet router attempts to validate the SIG(0) signature using that key. If the signature is valid, then the homenet router answers the query. If the zone is not signed, or doesn't have a trust anchor in the parent zone, the responding server signs the answer with its own SIG(0) key. The resolver that sent the query validates the response using DNSSEC if possible, and otherwise using the SIG(0) key.

[it can be argued that this isn't necessary for the base spec, and it obviously requires some additional protocol work, so may want to leave it out of the base architecture. It may also make more sense to serve queries using DNS-over-TLS (dprive) rather than SIG(0).]

4. Publishing the Public Namespace

4.1. Acquiring the Global Name

There are two ways to acquire a global name: the end-user can register a domain name using a public domain name registry, or the end-user can be assigned a subdomain of a registered domain by a homenet global name service provider. We will refer to this as the Global Name Registration Provider [GNRP]. In either case, the registration process can either be manual or automatic. Homenet routers support automatic registration regardless of the source of the homenet's global name, using a RESTful API.

The RESTful API provides a method for generating a unique URL which can be used for a limited time by the GNRP to register the global

name once the end user has chosen one and made payment arrangements (if necessary). When the GNRP is ready to convey the global name to the homenet, it uses the specified URL to submit (POST) the name. The URL will be https, but the certificate will not be valid; its purpose is to provide privacy, not authenticity.

In response, the homenet server either rejects the POST, if the URL has expired or is invalid, or else returns a text response containing a single label, '@', representing the local namespace. Under that label, the homenet will include one or more DNSKEY records for zone signing keys, one of which is required, and key signing keys, which are not required.

The returned zone will also include a TLSA record. The record has a Usage field value of 0 (PKI Cert), a Selector value of 1 (just the public key), and a matching type of 0 (exact match). The Certificate Association data field contains a public key generated by the homenet for use in authenticating local web traffic.

The returned zone may include NS records; if it does, the GNRP is expected to use those NS records in the delegation for the global name. Otherwise, it provides the NS records for its own authoritative servers.

The GNRP then sets up a secure delegation using the currently-valid ZSKs included in the zone. The GNRP also signs the public key provided in the TLSA record using a PKI cert owned by the GNRP that can be validated by web browsers, and posts it back to the homenet using the RESTful API URL.

More detail on this process will be provided in a future document.
[or really, how much detail should there be here?]

4.2. Hidden Primary/Public Secondaries

The default configuration for a homenet's external name service is that the primary server for the zone is not published in an NS record in the zone's delegation. Instead, the GNRP provides authoritative name service for the zone. Whenever the public zone is updated, the hidden primary sends NOTIFY messages to all the secondaries, using the zone's ZSK (or?) to sign the message.

When any of the GNRP secondary servers receives a notify for the zone, it checks to see that the notify is signed with a valid ZSK for that zone. If so, it contacts the IP address from which the NOTIFY was sent and initiates a zone transfer. Using this IP address avoids renumbering issues. Upon finishing the zone transfer, the zone is validated using each ZSK used to sign it. If any validation fails,

the new version of the zone is discarded. If updates have been received, but no valid updates received, over a user-settable interval defaulting to a day (or?), the GNRP will communicate to the registered user that there is a problem.

The reverse zone for any prefix delegated by an ISP should be delegated by that ISP to the home gateway to which the delegation was sent. The list of secondaries for that zone is sent to the home gateway using DHCPv6 prefix delegation (or?). The ZSK is announced to the ISP in each DHCP PD message sent by the home gateway. Whenever an update is made to this zone, the home gateway sends a NOTIFY to each of the listed secondaries for the delegation, and updates occur as described above.

4.3. DNSSEC security

All zones published by the homenet are signed. Internal zones cannot have secure delegations, however. Hosts that are aware of homenets can do TOFU authentication of a particular instance of the homenet zones [TBD1] or [TBD2]. To do this, the host queries the uuid.[TBD2] name. The homenet always publishes this name with a single TXT RR containing a UUID, which is expected to be unique and stable. The homenet will also publish a name, rev.[TBD2] which contains a PTR RRset that enumerates the outer delegations of all reverse zones operated by the homenet at the time of the query that are in private address spaces.

The homenet-aware host can then query and cache the ZSKs of the [TBD2] domain on that homenet, using the UUID to identify it. The homenet uses the same ZSK for all zones that it publishes. Homenet-aware hosts can validate any record in the [TBD1] and [TBD2] zones and in reverse zones for private and ULA number spaces using the stashed ZSK for the homenet UUID to which the host is currently connected (may be different on different interfaces). Names in non-private, non-ULA number spaces are validated using secure delegations, not homenet TOFU trust anchors, as are all other zones other than [TBD1] and [TBD2].

4.4. PKI security

PKI security is only possible if the homenet has a global name. The homenet should not use TLS unless it has a certificate that will be successfully validated by web servers; otherwise, the homenet will be training end-users to click through certificate warnings, and will be inoperable to web browsers with correct security user interfaces (which never present such warnings). If the homenet has a global name, it should also have gotten a valid PKI certificate as part of the process of acquiring the global name.

The homenet tracks the expiration date of the TLS certificate. One month before expiration, the homenet will send a renewal request to the GNRP using the URL provided by the GNRP during registration. The GNRP will then provide a new certificate and a new URL, which the homenet will record for the next renewal (the URL is not required to change). Because the key to be signed is published in the public namespace of the homenet, there is no need for a secondary authentication path for the key.

4.5. Renumbering

The homenet may renumber at any time. IP address RRs published in either namespace must never have a TTL that is longer than the valid lifetime for the prefix from which the IP address was allocated. If a particular ISP has deprecated a prefix (its preferred lifetime is zero), IP addresses derived from that prefix are not published in the DNS. If more than one prefix is provided by the same ISP and some have different valid lifetimes, only IP addresses in the prefix or prefixes with the longest valid lifetime are.

4.6. ULA

Question: If the homenet has one or more ULAs, should we only publish the ULAs and not the global addresses in the local namespace? This would prevent renumbering events from having any impact on local communication. Any reason not to do this? Would require some rewording of the local/global namespace text.

5. Management

5.1. End-user management

Need to have well-known name with RESTful API that apps can connect to, so that you can have an app on your phone, laptop or whatever that operates the network. This is a model that seems popular and accepted by end-users; having a well-defined API allows us to avoid a million different undocumented vendor-specific management APIs. Web API would also be nice, but we can't specify that, so better to specify the RESTful API and let vendors decide what sort of frock to put on it.

This API should provide a means for notifying end-users of issues on the home network, using whatever app they have installed. The API must provide a mechanism for registering end-users or devices that are permitted to manage the homenet, and a way to recover if all such devices are lost.

5.2. Central management

Possibly can be done mostly through RESTful API, but might want Netconf/Yang as well. Should be possible to have the local namespace mastered on an external DNS auth server, e.g. in case a bunch of HNRS are actually set up in an org, or in case an ISP wants to provide a service package for users who would rather not have an entirely self-operating network.

6. Privacy Considerations

Private information must not leak out as a result of publishing the public namespace. We believe the current provisions adequately address this concern. (right?)

7. Security Considerations

Need someone with security fu to review the registration model, etc., once we have it.

8. IANA considerations

IANA will add a new registry titled Homenet Management Well-Known Names, which initially contains:

uuid Universally Unique Identifier--TXT record containing, in base64 encoding, a stable, randomly generated identifier for the homenet that is statistically unlikely to be shared by any other homenet.

global-name The homenet's global name, represented as a PTR record to that name.

global-name-register The hostname of the homenet's global name registry service, with A and/or AAAA records.

all-resolver-names A list of all the names of the homenet's resolvers for the homenet PvD, represented as an RRset containing one or more PTR records.

The IANA will allocate two names out of the Special-use TLD names registry:

TBD1 Suggested value: "homenet"

TBD2 Suggested value: "_hnsd"

9. Normative References

- [1] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [2] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [3] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<http://www.rfc-editor.org/info/rfc2136>>.
- [4] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<http://www.rfc-editor.org/info/rfc6762>>.
- [5] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<http://www.rfc-editor.org/info/rfc6763>>.
- [6] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<http://www.rfc-editor.org/info/rfc7556>>.
- [7] Cheshire, S., "Hybrid Unicast/Multicast DNS-Based Service Discovery", draft-ietf-dnssd-hybrid-03 (work in progress), February 2016.
- [8] Korhonen, J., Krishnan, S., and S. Gundavelli, "Support for multiple provisioning domains in IPv6 Neighbor Discovery Protocol", draft-ietf-mif-mpvd-ndp-support-03 (work in progress), February 2016.

Author's Address

Ted Lemon
Nominum, Inc.
800 Bridge Parkway
Redwood City, California 94065
United States of America

Phone: +1 650 381 6000
Email: ted.lemon@nominum.com