Homenet Working Group
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

A. Augustin
Cisco Systems

July 6, 2015

DNCP Use Case in a Distributed Cache System
draft-augustin-homenet-dncp-use-case-00

Abstract

In a distributed cache system, at some point the nodes need to share and synchronise some information. This document explains how this cache system works and shows how DNCP is useful in this situation.

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 January 7, 2016.

Table of Contents

1. Introduction

The cache system described in this document works under the assumption that each chunk of data that may be cached is uniquely associated to an IPv6 address. When a client desires to retrieve this chunk of data, it establishes a TCP connection towards the IPv6 address of the chunk. The system only relies only on the use of IPv6 and TCP, it is therefore independant of the application used on top of it (usually HTTP). This connection is routed through a proxy, which acts as an entry point in the cache system. The proxy inserts a Segment Routing Header in each packet which destination is the IPv6 address of a chunk, in order to have these packets sequentially routed through a certain number of cache servers. Cache servers intercept the packets to reply directly to clients if they have the requested chunk available locally (Segment Routing interception).

2. Terminology and Abbreviations

The terminology and abbreviations used in this document are described in this section.

- DNCP: The Distributed Node Consensus Protocol, as defined in [I-D.ietf-homenet-dncp]
- SRH: IPv6 Segment Routing Header, defined in [I-D.previdi-6man-segment-routing-header]
- SR List: The list of addresses included in a SRH specifying the different destinations a packet will go through before reaching its final destination.
- Chunk: A blob of binary data, which is associated to a unique IPv6 address, and which a client may request.
- Client: A host that requests a chunk by establishing a connection towards the associated IPv6 address.

- Proxy: A node that insert SRHs in the packets corresponding to client requests in order to have them routed to cache servers.

- Cache server: A node that may have a certain number of chunks available. Upon receiving packets with a segment routing header, a cache server can either intercept the packets and reply directly if it has the requested chunk, or forward it to the next hop in the SRH.

- Source server: A node that has a given list of chunks available.

3. Context

The distributed cache system is composed of one or more proxies, cache servers, and source servers.

When a client needs a chunk of data, it establishes a TCP connection towards the IPv6 address identifying the chunk. This TCP connection is routed through a proxy which allows to keep compatibility with existing clients. Depending on the content being requested, which is known by looking at the destination address of the packet, the proxy chooses a SR List. The SR List contains, in this order, the addresses of several cache servers, the address of a source server that has this content, and the address of the chunk. This SR List is converted to a SRH and inserted in the packet. No information is retained per connection, this operation is applied to each packet individually.

The cache servers maintain a list of the chunks that they have cached and that are available locally. When receiving a packet, they look at the last address of the SRH. If the corresponding chunk is available, they intercept the packet and deliver it locally. If the chunk is not available, they forward it according to the SRH. This method ensures that the TCP connection is established with the first server in the SR List that has the requested chunk.

Each cache server monitors the requests that it receives by counting the TCP SYN packets and looking at the last address in the SRH. It caches the most requested chunks by using a caching algorithm, such as Least Frequently Used or Least Recently Used. This implies that, in order to be efficient, a given proxy should always use the same SR List for a given chunk.

From an other point of view, for a given SR List, the first server on the list caches as many chunks as possible among the most requested ones, the second server caches the most asked chunks that have not been cached by the first server, and so on. In order to minimize the distance that most requests cover in the network, proxies should

generate SR Lists that are dependant on the underlying topology: close servers should be put first in the SR List, then servers further away, and eventually a source server. Proxies should also balance the load between adjacent servers (clustering), stop using servers that are unavailable, and automatically start using new servers that become available.

As the servers are passive, the efficiency of the system will be dependant on how smartly the proxies generate SR Lists. The exact logic of the proxy is out of scope of this document, but the proxies don't need a lot of information to generate good SR Lists. They can deduce many things from:

- The list of proxies, cache servers and source servers that are available.
- The distances between all servers in the system, which can be measured from several metrics (for instance the RTT).

4. Interest of using DNCP

In this context, DNCP allows to extremely simply share the data needed (the list of nodes and the distances between them) across all nodes in the system, without reimplementing a full-fledged protocol. DNCP ensures that the data is synchronised between nodes and that data from unreachable nodes is removed.

The distance is measured at startup and is not republished unless it changes significantly. Consequently, the data only changes when a server becomes available or unavailable, or when something changes in the underlying network. This should not happen too often, and so the DNCP Network State Hash should not change too often, which will limit DNCP excitation.

Without DNCP, we would have needed to design a whole protocol with hello messages, keepalives, a flooding mechanism, and so on. DNCP manages all this automatically, all we had to do was to define which data had to be shared and adjust the profile for our needs (faster dead neighbor detection, unicast UDP, etc.).

5. DNCP Profile

This section describes the particular DNCP Profile used by the nodes in this system.

5.1. DNCP Constants and parameters

- The unicast transport protocol is UDP, no multicast transport protocol is used.
- The transport protocol is not secured.
- The UDP port used is 16255.
- Upon node identifier collision, both nodes will choose a new random node identifier.
- Imin = 0.1s, Imax=20s, K=1
- The non cryptographic hash function used is MD5.
- DNCP_NODE_IDENTIFIER_LENGTH = 4
- DNCP_GRACE_INTERVAL = 30s
- Keepalives are enabled, DNCP_KEEPALIVE_INTERVAL = 5s, DNCP_KEEPALIVE_MULTIPLIER = 2.1

5.2. DCNP TLVs used

5.2.1. Server Announce TLV

This TLV is published when a server is ready to join the system. It includes the node type: cache, source server or proxy; and the node’s IPv6 addresses.

5.2.2. Server Distance TLV

Whenever a node receives a new Server Announce TLV, it measures the distance to the new server (for instance by pinging it), and then publish a Server Distance TLV. This TLV includes the distance and the public IPv6 of the pinged server.

5.3. DNCP Setup

Because this system is to be inserted in a ISP network, we do not assume that a multicast routing protocol is running. Because of this, we use only unicast communications.

The system includes one or more nodes called "DNCP Relays" that have fixed addresses. Other nodes are configured with the addresses of these relays and contact them in unicast on startup. When they are ready, the nodes publish their Server Announce TLV, and start sending Server Distance TLVs for each Server Announce TLV they receive.

By extracting the data from DNCP, the proxies can get an idea of where each server is, and optimize the SR Lists they generate accordingly. Moreover, as they know that other proxies are using exactly the same data, they can know what their decisions will be and collaborate to a certain extent without exchanging additionnal data.

6. IANA Considerations

This draft includes no requests to IANA.

7. Security Considerations

No security considerations.

8. References

8.1. Normative References

[I-D.ietf-homenet-dncp]
Stenberg, M. and S. Barth, "Distributed Node Consensus Protocol", draft-ietf-homenet-dncp-05 (work in progress), June 2015.

8.2. Informative references

[I-D.previdi-6man-segment-routing-header]
Previdi, S., Filsfils, C., Field, B., and I. Leung, "IPv6 Segment Routing Header (SRH)", draft-previdi-6man-segment-routing-header-06 (work in progress), May 2015.

Author's Address

Aloys Augustin
Cisco Systems

Email: aloys.augustin@polytechnique.org

Homenet Working Group L. Geng
Internet-Draft H. Deng
Intended status: Standards Track China Mobile
Expires: January 4, 2016 July 3, 2015

Use Cases for Multiple Provisioning Domain in Homenet
draft-geng-homenet-mpvd-use-cases-01

Abstract

This document describes the use cases of multiple provisioning domain (MPVD) in homenet. Although most residential networks nowadays are connected to a single ISP and normally subscribed to standard internet service, it is expected that much wider range of devices and services will become common in home networks. Homenet defines such home network topologies with increasing number of devices with the assumption that it requires minimum configuration by residential user. As described in the homenet architecture ([RFC7368]), multihoming and multi-service residential network will be more common in the near future. Nodes in such network may commonly have multiple interfaces or subscribe to multiple services. Potential types of PVD-aware nodes concerning interface and service specific provisioning domains are introduced in this document. Based on this, different MPVD configuration examples are given. These examples illustrate how PVD may be implemented in home network. PVDs provide independent provisioning domains for different interfaces and services, which enables robust and flexible network configuration for these networks.

Table of Contents

1. Introduction

It is believed that future residential network will more commonly be multihomed, which potentially provides either resilience or more flexible services. At the same time, more internal routing and multiple subnets are expected to commonly exist in such networks. For example, customer may want independent subnets for private and guest usages. Homenet describes such future home network involving multiple routers and subnets ([RFC7368]).

Multihoming and the increasing number of subnets bring challenges on provisioning of the network. As stated in [RFC6418], such multihomed scenarios with nodes attached to multiple upstream networks may experience configuration conflicts, leading to a number of problems. To deal with these problem, draft-ietf-mif-mpvd-arch-10 provides a framework which introduces Provisioning Domain (PvD), which associates a certain interface and its related network configuration information. Hence, corresponding network configuration can be used when packets are delivered through a particular interface.

This document focuses on the MPvD use cases in residential network, particularly the IPV6-based homenet. Based on the homenet topology, use cases of MPvD in homenet are described for both singlehomed and multihomed network configurations.

1.1. 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 RFC 2119 [RFC2119].

2. Terminology and Abbreviations

The terminology and abbreviations used in this document are defined in this section.

- ISP: Internet Service Provider. A traditional network operator who provides internet access to customers.

- VSP: Virtual Service Provider. An service provider who typically provides over-the-top services including but not limited to Internet of Things services (IoT).

3. Homenet with Multiple PvDs

In the most common multihoming scenarios, the home network has multiple physical connections to the ISP networks. Section 3.2.2.2 and 3.2.2.3 in [RFC7368] give the topology examples of such homenet. In the examples, homenet hosts are connected to a single or multiple customer edge routers (CE router), the CE routers are then connected to separate ISP networks. For the particular topology with a single CE router given in Section 3.2.2.3 in [RFC7368], the CE router is a mif node since it has two interfaces connected to individual service provider routers. Given that the CE router is a PvD-aware node, it may have a single PvD as it is connected to only one ISP and an additional PvD if connected to both.

Apart from the multihoming resulted from physical connections to different ISPs, the future residential network may also logically connected to multiple Over-the-top service providers(i.e. IoT service providers), who do not directly provide internet access service to customers. For example, one customer may subscribe to a traditional service ISP for basic internet service, whilst subscribe to other providers for Internet of Things service. The latter are likely to be VSPs as defined in Section 2 of this document, who are not bounded to any of the ISPs providing basic access service for the residential network. In this case, a particular VSP may also use PvDs for customized network configurations purposes. This enables the VSPs to provide independent and flexible provisioning between on its subscriber's network for different services. Meanwhile, VSPs may also want to use independent PvDs to avoid the configuration conflicts between each other as stated in RFC6418.

The following sections outline differenct types of PvD in homenet.

3.1. PvD associating an interface in homenet

One typical example of a PvD in home network is the one associating an interface with network configuration. As described in ([RFC7368]), a multihomed CE router in homnet connects with multiple ISPs. It may have several different uplink interfaces (i.e. PON and LTE). ISP can use PvDs to associate its network configurations with specific uplink interface the CE router provides to its subscriber. PvD information can be delivered by ISP through the corresponding uplink interface of the CE router.

Another scenario is when the interior routers and hosts in homenet have multiple uplink interfaces(i.e. LAN, WIFI, Zigbee). Depending on the actual network implementation, PvDs for these interfaces can be either configured by ISPs or VSPs. Since these devices connect with the internet through the CE router within homnnet, mechanism needs to be defined for PvD information delivery within homenet.

3.2. PvD associating a service in homenet

This type of PvD is useful in homenet when a provisioning domain is needed for a specific service that a homenet user subscribe to. Unlike the PvD associating an interface in homenet, this PvD associates the network configuration with a service. The service can be provided by any ISP or VSP that is available to the user.

In homenet, ISPs and VSPs can use this PvD for service-specific provisioning in CE routers, interior routers and hosts. Service providers could decide the number of PvDs they offer depending on the services a user subscribes to. This enables complete dependency

between the provisioning of different services provided by either same or different sources.

A good example of a device in homenet that may use such PvD is a IoT box provided by VSP. This box may be connected to a CE router as an interior router as demonstrated in section 3.2.2.1 of [RFC7368], or integrated with the CE router or the host in some circumstances.

Since some services may need provisioning domain in multiple devices in homenet(i.e. Both IoT service gateway and host need to be provisioned), PvDs associated with the same service in different devices should ideally be managed by a single provider. Hence, necessary collaboration can be taken care of by the VSP/ISP.

3.3. PvD for hybrid purposes in home network

The coexistence of multiple interfaces and services is possible when a device in homenet is both multihomed with more than one ISPs and subscribed to multiple services. Assuming that a service provider has access to the interface of the device on which its service is provided, a provisioning domain associating both the service and corresponding interface can be used for a simpler set-up. For example, instead of separately maintaining a WIFI interface PvD and an IoT service PvD, an VSP can merge the information of these to PvD and only use a single PvD for hybrid provisioning. It is always preferred that PvD for hybrid provisioning is used when possible, since it offers better network configuration flexibility for service provider and enables seamless coordination between interface and the service runs on it.

4. Examples of MPvD Configurations in Home Network

This section gives some examples of MPvD configurations in home network.

4.1. Homenet Connected to a Single ISP

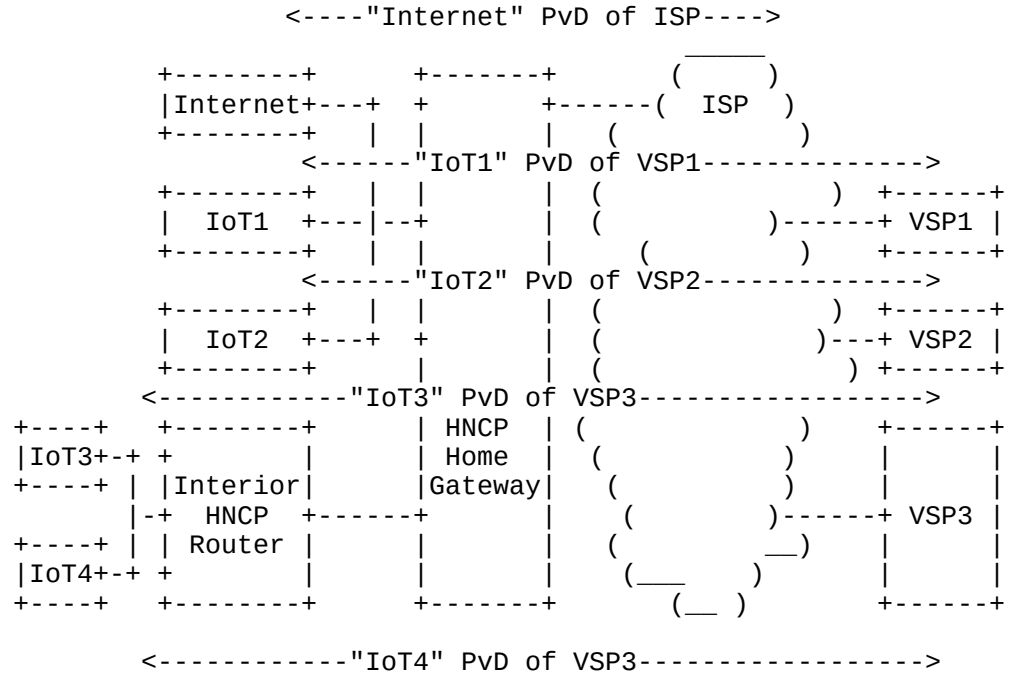


Figure 1

A homenet home gateway (CE router) is singlehomed with a single ISP as seen in Figure 1. In this scenario, basic internet service is provided by this ISP, IoT services are provided by 3 different VSPs. Multiple PvDs are created for the CE router for the purpose of service provisioning. The home gateway, connected with multiple service providers, receives basic internet PvD information from the connected ISP, IoT1 PvD information from VSP1 and IoT2 PvD information from VSP2.

Additionally, an HNCP-enabled interior router is connected to the home gateway and provides another 2 IoT services from VSP3 to the user. VSP3 may create 2 PvDs associating IoT3 and IoT4 respectively for the interior router, subject to the user’s subscription.

In this example, ISP provides basic internet service through a homenet home gateway and VSPs provides multiple IoT services through both the HNCP home gateway and the interior HNCP router. Since none of the gateway devices are multihomed, all of the PvDs associate corresponding network configuration with a specific service. The PvD information is delivered by ISP and VSPs through the corresponding singlehomed interface.

4.2. Multihomed Homenet

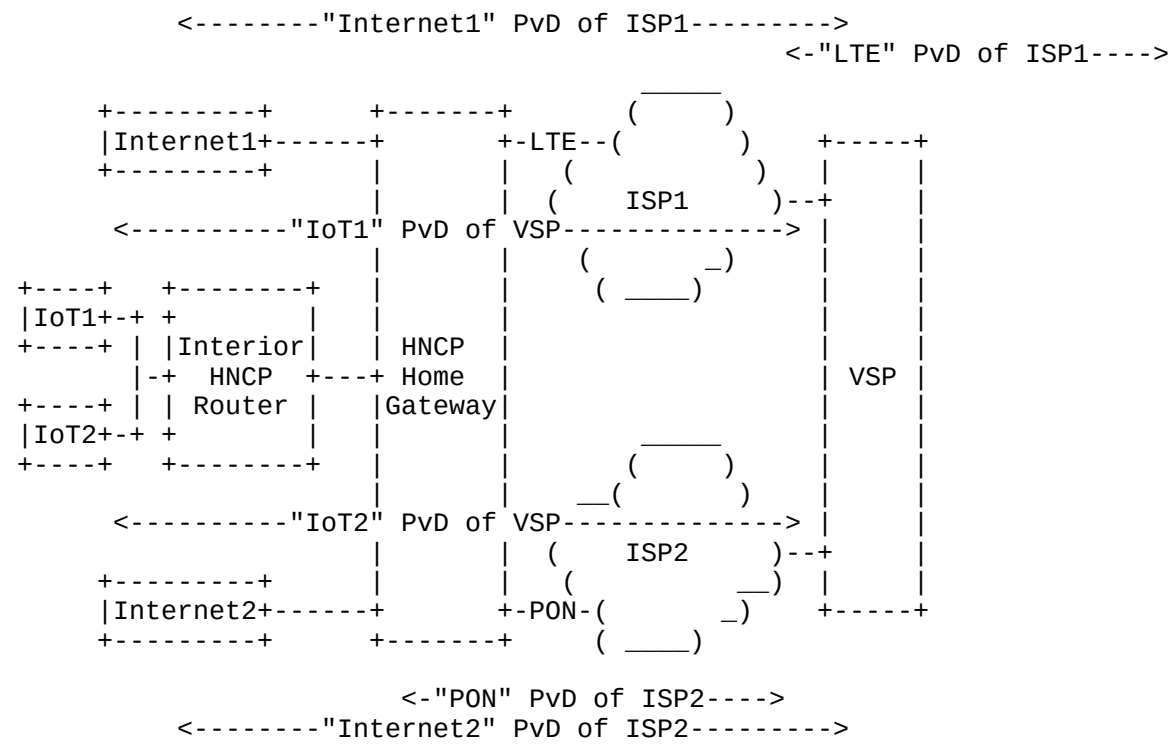


Figure 2

Figure 2 illustrates an example of multihomed HNCP home gateway. In this example, two ISPs are connected with the HNCP home gateway and both provide internet services. The interfaces used by the HNCP home gateway for these 2 ISPs are LTE and PON, which are provisioned within the LTE PvD of ISP1 and PON PvD of ISP2. Another 2 PvD for individual internet service are also created by ISP1 and ISP2 respectively. In principle, it is preferred in this example that the 2 PvDs of the same ISP should be merged as one hybrid PvD, which associates the complete set of network configuration with both the internet service and the corresponding uplink interface.

The interior HNCP router in this example are similiar to the one in the previous example, providing 2 IoT services provisioned separately by VSP. However, it is worth mentioning that the IoT1 and IoT2 PvDs information is not necessarily delivered from a fixed ISP because of

the nature of multihomed gateway. It is upto the VSP to make the decision according to the available network resources.

Although not shown in Figure 2, the HNCP home gateway may also directly provide IoT service from VSP. Since VSP is normally homed with multiple ISPs, the multihomed HNCP home gateway may receive PvD information from different ISPs for the IoT service. PvD configuration conflicts need to be avoided in this case.

5. PvD-aware node in homenet

PvD-aware node is the device where the provisioning domains and associated network configuration are set up. In the examples given in Section 4, the HNCP home gateways and Interior HNCP routers are all PvD-aware nodes. The HNCP home gateway receives MPvD identity and associated network configuration from the network and forward the MPvD information belonging to interior routers. It is worth mentioning that a PvD-aware node may also be a host in homenet,which is not shown in the given examples. For instance, a mobile device connected with the interior router may have MPvDs for it’s Wifi and cellular interfaces, or for multiple services it subscribed to.

As introduced in RFC7556, typical information a PvD-aware node learned from network by a provisioning domain including source address prefixes for use by the connected hosts within the PvD,IP address(es) of the DNS server(s) and default gateway address. While these information is maintained in the PvD-aware node, it needs to assign prefixes to the connected hosts within homenet using homenet-defined approaches.

6. Conveying PvD information

The PvD associated with an VSP service may be directly provided by VSP using application layer approach, or provided by the the ISPs in which the VSP is homed if there are collaboration between ISPs and VSPs.

At the time this document was written, the conveying of PvD information was still under discussion in mif working group. Popular choices include DHCP and Route Advertisement. For PvD information provided from ISPs and VSPs to the CE routers, the approaches for PvD information delivery defined by mif may be directly used. For PvD information delivery within homenet between HNCP-enabled routers and hosts, HNCP-related approach need to be concerned. The detail of how homenet could support the delivery of PvD information is subjected to further discussions and will be addressed in a separate document.

7. Acknowledgements

The author would like to thank Ted Lemon for valuable initial discussions of this document.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

TBA

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

10.2. Informative References

[RFC6418] Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", RFC 6418, November 2011.

[RFC7368] Chown, T., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", RFC 7368, October 2014.

Authors' Addresses

Liang Geng
China Mobile

Email: gengliang@chinamobile.com

Hui Deng
China Mobile

Email: denghui@chinamobile.com

Homenet Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2016

M. Stenberg

S. Barth

July 3, 2015

Distributed Node Consensus Protocol
draft-ietf-homenet-dncp-07

Abstract

This document describes the Distributed Node Consensus Protocol (DNCP), a generic state synchronization protocol which uses Trickle and Merkle trees. DNCP leaves some details unspecified or provides alternative options. Therefore, only profiles which specify those missing parts define actual implementable DNCP-based protocols.

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 January 4, 2016.

Table of Contents

1. Introduction

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

For state synchronization a Merkle tree is used. It is formed by first calculating a hash for the dataset, called node data, published by each node, 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) 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 neighbors are present, transitively, the whole network state also stays up-to-date.

DNCP is most suitable for data that changes only infrequently to gain the maximum benefit from using Trickle. As the network of nodes, or the rate of data changes grows over a given time interval, 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. 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.

2. Terminology

DNCP profile	a definition of the set of rules and values defining the behavior of a fully specified, implementable protocol which uses DNCP. The DNCP profile specifies transport method to be used, which optional parts of the DNCP specification are required by that particular protocol, and various parameters and optional behaviors. In this document any parameter that a DNCP profile specifies is prefixed with DNCP_. Contents of a DNCP profile are specified in Section 9.
DNCP-based protocol	a protocol which provides a DNCP profile, and potentially much more, e.g., protocol-specific TLVs and guidance on how they should be used.
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 the same DNCP-based protocol. 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	As DNCP itself is relatively transport agnostic, an address in this specification denotes just something that identifies an endpoint used by the transport protocol employed by a DNCP-based protocol. In case of an IPv6 UDP transport, an address in this specification refers to a tuple (IPv6 address, UDP port).
Endpoint	a locally configured communication endpoint of a DNCP node, such as a network socket. It is either bound to an Interface for multicast and unicast communication, or configured for explicit unicast communication with a predefined set of remote addresses. Endpoints are usually in one of the transport modes specified in Section 4.2.

Endpoint identifier	a 32-bit opaque 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 a 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.
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. 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.

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 discovers the topology of its nodes and maintains the liveliness of published node data by ensuring that the publishing node was - at least recently - bidirectionally reachable. This is determined, e.g., by a recent and consistent multicast or unicast TLV exchange with its peers. 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 a later specification.

A Merkle tree is maintained by each node to represent the state of all currently reachable nodes and the Trickle algorithm is used to trigger synchronization. The need to check neighboring nodes for state changes is thereby determined by comparing the current root of their respective trees, i.e., their individually calculated network state hashes.

Before joining a DNCP network, a node starts with a Merkle tree (and therefore a calculated network state hash) only consisting of the node itself. It then announces said hash 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 Merkle 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 Merkle 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. Merkle Tree

Each DNCP node maintains a Merkle tree of height 1 to manage state updates of individual DNCP nodes, the leaves of the tree, and the network as a whole, the root of the tree.

Each leaf represents one recently bidirectionally reachable DNCP node (see Section 4.6), and is represented by a tuple consisting of the node's sequence number in network byte order concatenated with the hash-value of the node's ordered node data published in the Node State TLV (Section 7.2.3). These leaves are ordered in ascending order of the respective node identifiers. The root of the tree - the network state hash - is represented by the hash-value calculated over all such leaf tuples concatenated in order. It is used to determine whether the view of the network of two or more nodes is consistent and shared.

The leaves and the root network state hash are updated on-demand and whenever any locally stored per-node state changes. This includes local unidirectional reachability encoded in the published Neighbor TLVs (Section 7.3.2) and - when combined with remote data - results in awareness of bidirectional reachability changes.

4.2. Data Transport

DNCP has relatively 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. A specific definition of the transport(s) in use and their parameters MUST be provided by the DNCP profile.

TLVs 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. TLVs in general are handled individually and statelessly, 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 an 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:

- If using a stream transport, the TLV MUST be sent at least once, and it SHOULD be sent only once.

- 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 speeds 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 employed, Trickle is not used at all. Where Trickle reset has been specified, a single Network State TLV (Section 7.2.2) is sent instead to every unicast peer. Additionally, recently changed Node State TLVs (Section 7.2.3) MAY be included.

  * If only unreliable unicast transport is employed, Trickle state is kept per each peer and it is used to send Network State TLVs every now and then, 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 every now and then, 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 broadcast link optimization (Section 6.2).

4.3. Trickle-Driven Status Updates

The Trickle algorithm has 3 parameters: Imin, Imax and k. Imin and Imax represent the minimum and maximum values for I, which 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 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. 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:

- the endpoint is in Multicast+Unicast transport mode, in which case the TLV MUST be sent over multicast.
- 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 also 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) over unicast to the originator of the TLV being processed. If the TLV being replied to was received via multicast and it was sent to a link with shared bandwidth, the reply SHOULD be delayed by a random timespan in [0, Imin/2], to avoid potential simultaneous replies that may cause problems on some links. Sending of replies MAY also be rate-limited or omitted for a short period of time by an implementation. However, an implementation MUST eventually reply to similar repeated requests, as otherwise state synchronization breaks.

A DNCP node MUST process TLVs received from any valid 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 neighbor 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:

- 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 MUST NOT contain the optional node data part unless explicitly specified in the DNCP profile.

- 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.

- 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 Imin. 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 Imin. 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.

- 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 an sequence number significantly (e.g., 1000) greater than the received one, to reclaim the node identifier. 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 H(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 <=> (a - b) % 2^32 & 2^31 != 0 is RECOMMENDED unless the DNCP profile defines another.

o Any other TLV: TLVs not recognized by the receiver MUST be silently ignored.

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

4.5. Adding and Removing Peers

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 Neighbor TLV (Section 7.3.2) MUST be created for it.

- 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 employed to ensure its presence. When the peer is no longer present, the Neighbor TLV and the local DNCP peer state MUST be removed.

If the local endpoint is in the Multicast-Listen+Unicast transport mode, a Neighbor TLV (Section 7.3.2) MUST NOT be published for the peers not having the highest node identifier.

4.6. Data Liveliness Validation

The topology graph MUST be traversed either immediately or with a small delay shorter than the DNCP profile-defined Trickle Imin, whenever:

- A Neighbor TLV or a whole node is added or removed, or
- the origination time (in milliseconds) of some node’s node data is less than current time - 2^32 + 2^15.

The topology graph traversal starts with the local node marked as reachable. 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:

- The origination time (in milliseconds) of R’s node data is greater than current time in - 2^32 + 2^15.
- R publishes a Neighbor TLV with:
  - Neighbor Node Identifier = N’s node identifier
  - Neighbor Endpoint Identifier = NE’s endpoint identifier
  - Endpoint Identifier = RE’s endpoint identifier
- N publishes a Neighbor TLV with:
  - Neighbor Node Identifier = R’s node identifier
  - Neighbor 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 removed 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 and to reduce the number of redundant state transmissions between nodes.

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:

- 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:

- 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.

- 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.

- 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.

- 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:

- Endpoint identifier: the 32-bit opaque value uniquely identifying it within the local node.
- 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:

- Interface: the assigned local network interface.
- Unicast address: the DNCP node it should connect with.
- Range of addresses: the DNCP nodes that are allowed to connect.

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

- Node identifier: the unique identifier of the peer.
- Endpoint identifier: the unique endpoint identifier used by the peer.
- Peer address: the most recently used address of the peer (authenticated and authorized, if security is enabled).
- 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

Trickle-driven status updates (Section 4.3) provide a mechanism for handling of new peer detection on an endpoint, 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.

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 or per-peer 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_KEEPALIVE_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 7.3.3) 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:

- 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:

- 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. When adding a new peer, it is initialized to the current time.

- 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 transmission time 't' in [I/2, I] MUST be randomly chosen. The actual sending time SHOULD be further delayed by a random timespan in [0, Imin/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 transmission time 't' in [I/2, I] MUST be randomly chosen.

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.

6.1.5. Neighbor Removal

For every peer on every endpoint, the endpoint-specific keep-alive interval must be calculated by looking for Keep-Alive Interval TLVs (Section 7.3.3) 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 Neighbor TLV for that peer and the local DNCP peer state MUST be removed.

6.2. Support For Dense Broadcast Links

This optimization is needed to avoid a state space explosion. Given a large set of DNCP nodes publishing data on an endpoint that actually uses multicast on a link, every node will add a Neighbor TLV (Section 7.3.2) 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 employed, there will be O(N^2) keep-alives running on the link if liveliness 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 neighbors 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. Main consideration when selecting a

bound (if any) for a particular type of link should be whether it supports broadcast traffic, and whether a too large number of neighbors 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 employed, 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. 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 normally form peer relationships with all peers.

6.3. Node Data Fragmentation

A DNCP-based protocol may be required to support node data which would not fit the maximum size of a single Node State TLV (Section 7.2.3) (roughly 64KB of payload), or use a datagram-only transport with a limited MTU and no reliable support for fragmentation. To handle such cases, a DNCP profile MAY specify a fixed number of trailing bytes in the node identifier to represent a fragment number indicating a part of a node’s node data. The profile MAY also specify an upper bound for the size of a single fragment to accommodate limitations of links in the network. Note that the maximum size of fragment also constrains the maximum size of a single TLV published by a node.

The data within Node State TLVs of all fragments MUST be valid, as specified in Section 7.2.3. The locally used node data for a particular node MUST be produced by concatenating node data in each fragment, in ascending fragment number order. The locally used

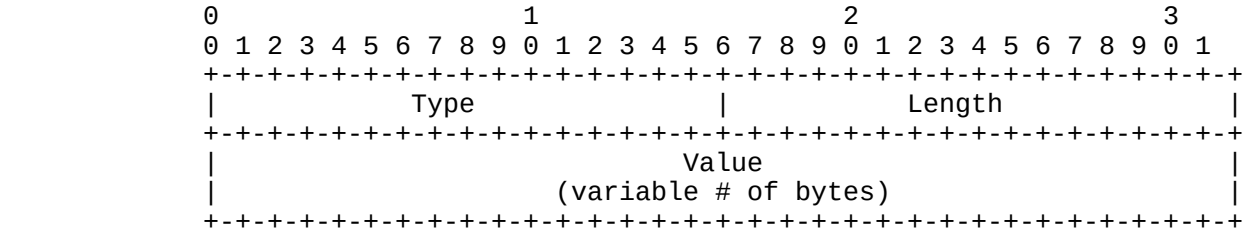
concatenated node data MUST still follow the ordering described in Section 7.2.3.

Any transmitted node identifiers used to identify the own or any other node MUST have the fragment number 0. For algorithm purposes, the relative time since the most recent fragment change MUST be used, regardless of fragment number. Therefore, even if just some of the node data fragments change, they all are considered refreshed if one of them is.

If using fragmentation, the data liveliness validation defined in Section 4.6 is extended so that if a Fragment Count TLV (Section 7.3.1) is present within the fragment number 0, all fragments up to fragment number specified in the Count field are also considered reachable if the fragment number 0 itself is reachable based on graph traversal.

7. Type-Length-Value Objects

Each TLV is encoded as a 2 byte type field, followed by a 2 byte length field (of the value excluding header, in bytes, 0 meaning no value) followed by the value itself, if any. Both type and length fields in the header as well as all integer fields inside the value - unless explicitly stated otherwise - are represented in network byte order. Padding bytes with value zero MUST be added up to the next 4 byte boundary if the length is not divisible by 4. These padding bytes MUST NOT be included in the number stored in the length field.



For example, type=123 (0x7b) TLV with value ’x’ (120 = 0x78) is encoded as: 007B 0001 7800 0000.

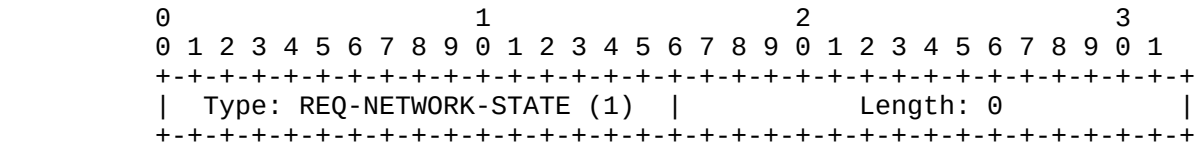
In this section, the following special notation is used:

.. = octet string concatenation operation.

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

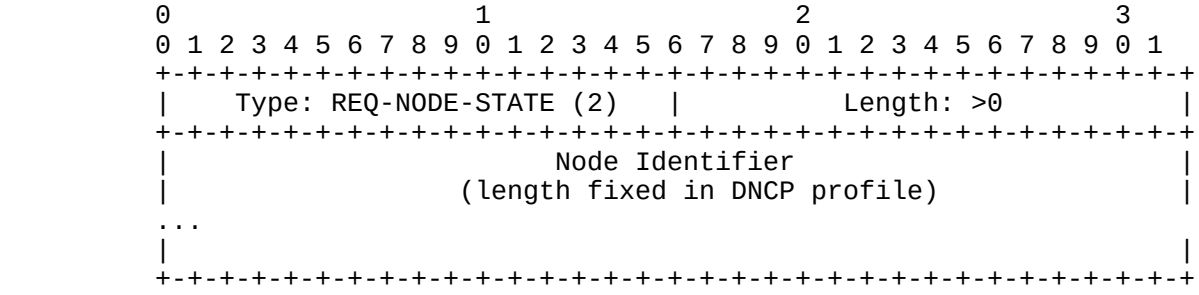
7.1. Request TLVs

7.1.1. Request Network State TLV



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).

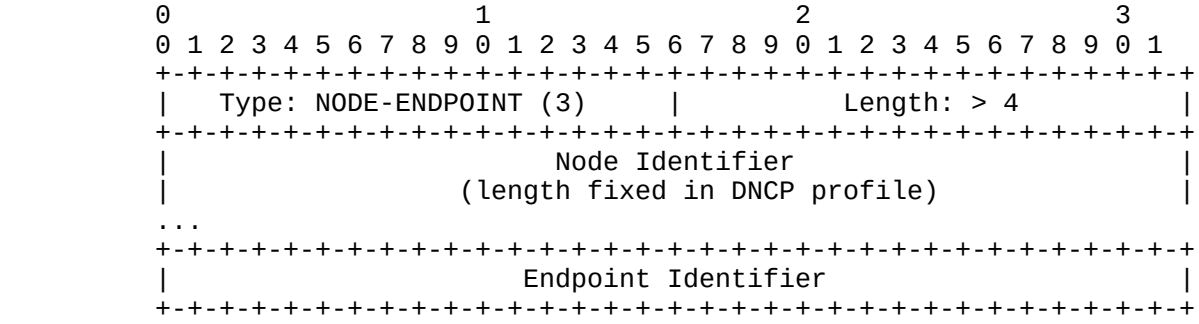
7.1.2. Request Node State TLV



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.

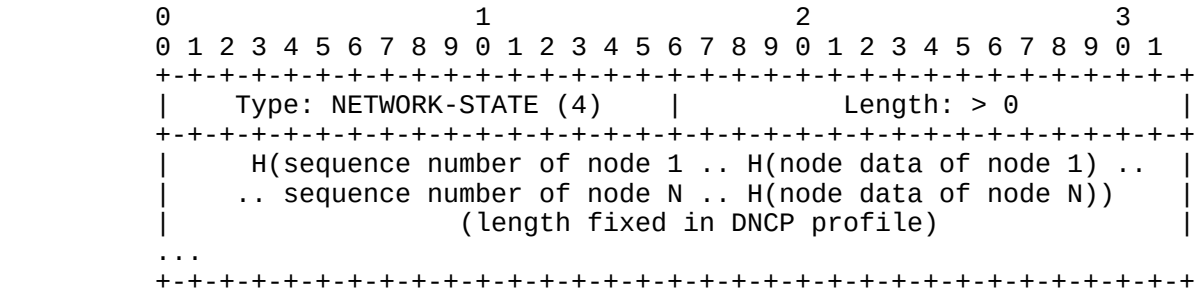
7.2. Data TLVs

7.2.1. Node Endpoint TLV



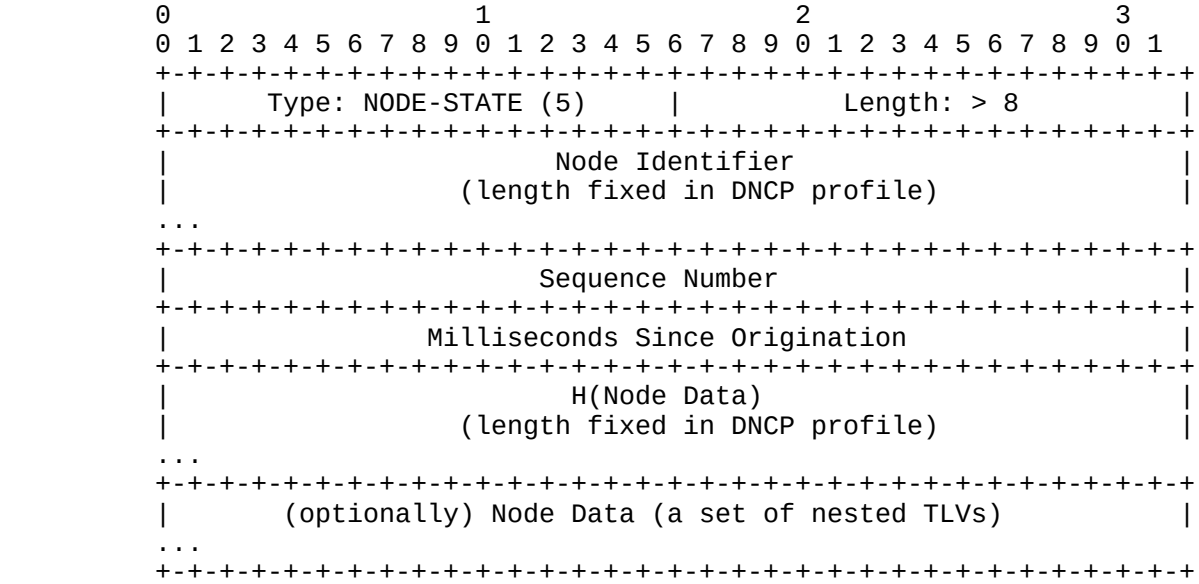
This TLV identifies both the local node’s node identifier, as well as the particular endpoint’s endpoint identifier.

7.2.2. Network State TLV



This TLV contains the current locally calculated network state hash, see Section 4.1 for how it is calculated.

7.2.3. Node State TLV



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 originating one, 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.

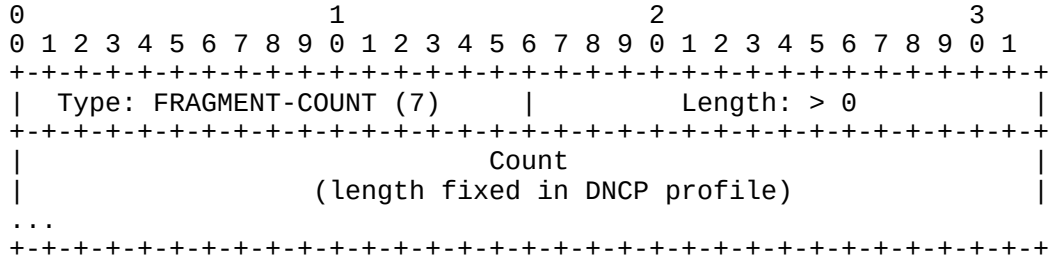
Absent any changes, if the originating node notices that the 32-bit milliseconds since origination value would be close to overflow (greater than 2^32-2^16), the node MUST re-publish its TLVs even if there is no change. In other words, absent any other changes, the TLV set MUST be re-published roughly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

The actual node data of the node may be included within the TLV as well in the optional Node Data field. In a DNCP profile which supports fragmentation, described in Section 6.3, the TLV data may be only partial but it MUST contain full individual TLVs. 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(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

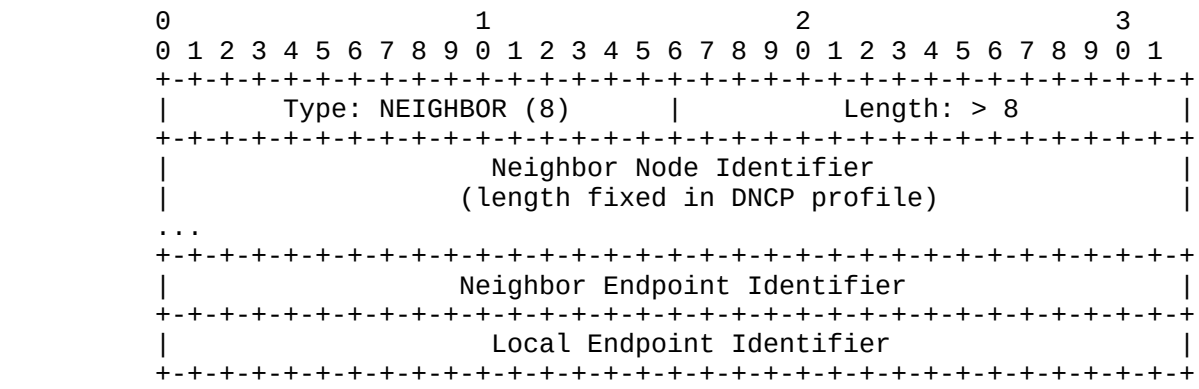
These TLVs are published by the DNCP nodes, and therefore only encoded within the Node State TLVs. If encountered outside Node State TLV, they MUST be silently ignored.

7.3.1. Fragment Count TLV



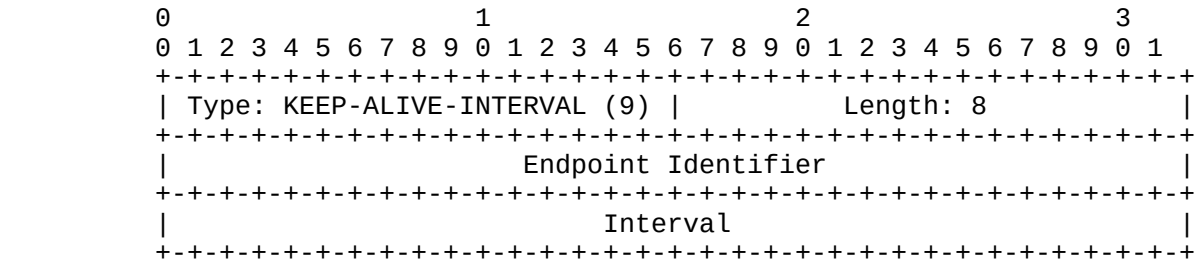
If the DNCP profile supports node data fragmentation as specified in Section 6.3, this TLV indicates that the node data is encoded as a sequence of Node State TLVs. Following Node State TLVs with Node Identifiers up to Count greater than the current one MUST be considered reachable and part of the same logical set of node data that this TLV is within. The fragment portion of the Node Identifier of the Node State TLV this TLV appears in MUST be zero.

7.3.2. Neighbor TLV



This TLV indicates that the node in question vouches that the specified neighbor 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 neighbor recently. For guaranteed up-to-date bidirectional reachability, the existence of both nodes' matching Neighbor TLVs needs to be checked.

7.3.3. Keep-Alive Interval TLV



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 endpoint for which the interval applies. 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

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:

- 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.
- 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.
- 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.

8.3.3. Announcement of Verdicts

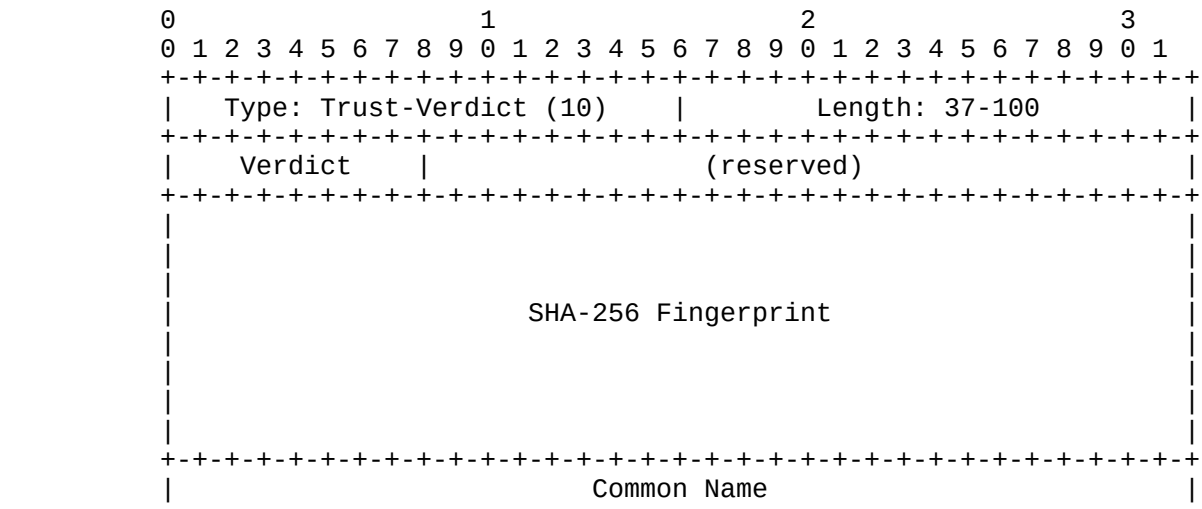
A node SHOULD always announce any configured trust verdicts it has established by itself, and it MUST do so if announcing the configured trust verdict leads to a change in the current effective trust verdict for the respective certificate. In absence of configured verdicts, it MUST announce cached trust verdicts it has stored in its trust cache, if one of the following conditions applies:

- The stored trust verdict is Cached Trust and the current effective trust verdict for the certificate is Neutral or does not exist.
- The stored trust verdict is Cached Distrust and the current effective trust verdict for the certificate is Cached Trust.

A node rechecks these conditions whenever it detects changes of announced trust verdicts anywhere in the network.

Upon encountering a node with a hierarchy of certificates for which there is no effective trust verdict, a node adds a Neutral Trust-Verdict-TLV to its node data for all certificates found in the hierarchy, and publishes it until an effective trust verdict different from Neutral can be found for any of the certificates, or a reasonable amount of time (10 minutes is suggested) with no reaction and no further authentication attempts has passed. Such trust verdicts SHOULD also be limited in rate and number to prevent denial-of-service attacks.

Trust verdicts are announced using Trust-Verdict TLVs:



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. Final byte MUST have value of 0.

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 neighboring 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:

- 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.

- 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.

- 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.

- When receiving TLVs, what sort of TLVs are ignored in addition - as specified in Section 4.4 - e.g., for security reasons. A DNCP profile may safely 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.

- 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.

- 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.

- 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 specific data, and network state hash, which is a hash of node specific data hashes. SHA-256 defined in [RFC6234] is the recommended default choice, but a non-cryptographic hash function could be used as well.

- DNCP_NODE_IDENTIFIER_LENGTH: The fixed length of a node identifier (in bytes).

- Whether to send keep-alives, and if so, whether per-endpoint (requires multicast transport), or per-peer. Keep-alive has also associated parameters:

    * DNCP_KEEPALIVE_INTERVAL: How often keep-alives are to be sent by default (if enabled).

    * DNCP_KEEPALIVE_MULTIPLIER: How many times the DNCP_KEEPALIVE_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.

- Whether to support fragmentation, and if so, the number of bytes reserved for fragment count in the node identifier.

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 should 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 Certifcate 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 DNCP TLV types, with the following initial contents:

0: Reserved

1: Request network state

2: Request node state

3: Node endpoint

4: Network state

5: Node state

6: Reserved (was: Custom)

7: Fragment count

8: Neighbor

9: Keep-alive interval

10: Trust-Verdict

32-191: Reserved for per-DNCP profile use

192-255: Reserved for per-implementation experimentation. How collision is avoided is out of scope of this document.

For the rest of the values (11-31, 256-65535), policy of 'standards action' should be used.

12. References

12.1. Normative references

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, March 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

12.2. Informative references

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

[RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

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

Q: 32-bit endpoint id?

A: Here, it would save 32 bits per neighbor 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 C. Changelog [RFC Editor: please remove]

draft-ietf-homenet-dncp-06:

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

draft-ietf-homenet-dncp-04:

- 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:

- Renamed connection -> endpoint.
- !!! 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:

- Changed DNCP "messages" into series of TLV streams, allowing optimized round-trip saving synchronization.
- Added fragmentation support for bigger node data and for chunking in absence of reliable L2 and L3 fragmentation.

draft-ietf-homenet-dncp-01:

- 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.
- Facilitated (simple) read-only clients by making Node Connection TLV optional if just using DNCP for read-only purposes.
- 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:

- TLVs were renumbered.
- 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).
- 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.

- Instead of 'ping', use 'keep-alive' (optional) for dead peer detection. Different message used!

Appendix D. 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 E. Acknowledgements

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

Authors' Addresses

Markus Stenberg
Helsinki 00930
Finland

Email: markus.stenberg@iki.fi

Steven Barth
Halle 06114
Germany

Email: cyrus@openwrt.org

HOMENET
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2016

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

W. Cloetens
SoftAtHome
July 2, 2015

Outsourcing Home Network Authoritative Naming Service
draft-ietf-homenet-front-end-naming-delegation-03.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

Copyright Notice

Table of Contents

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.

3. Terminology

- Customer Premises Equipment: (CPE) is the 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.

- 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.

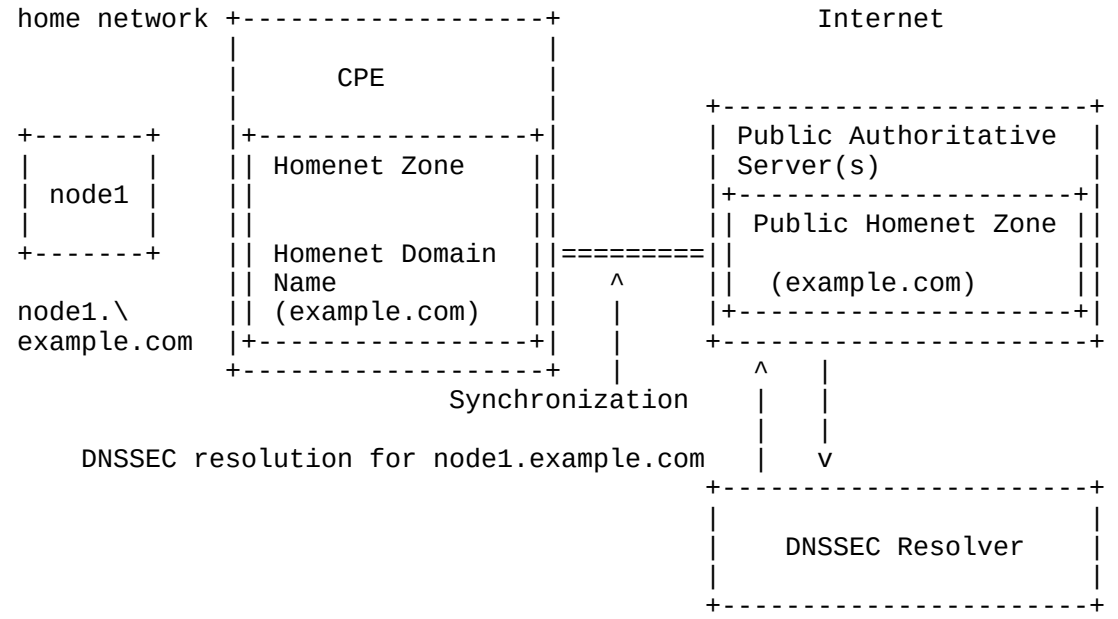


Figure 1: Homenet Naming Architecture Description

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.

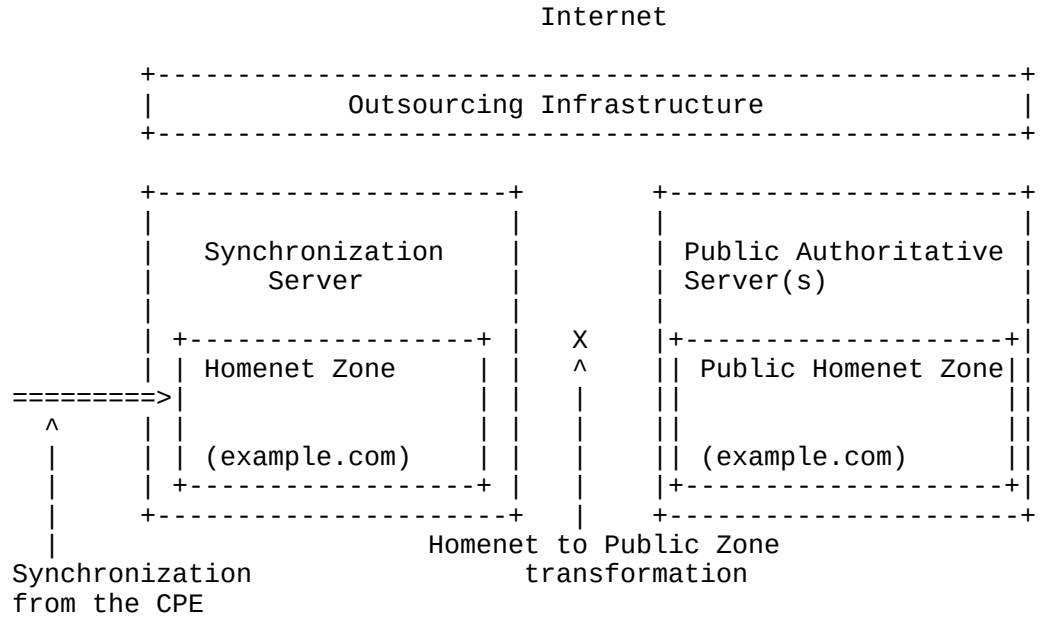


Figure 2: Outsourcing Infrastructure Description

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.

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 [I-D.ietf-dnssd-requirements].

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 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 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, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, August 1996.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, August 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

[RFC2142] Crocker, D., "MAILBOX NAMES FOR COMMON SERVICES, ROLES AND FUNCTIONS", RFC 2142, May 1997.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, May 2000.

[RFC2930] Eastlake, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, September 2000.

[RFC2931] Eastlake, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, September 2000.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, June 2006.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5936] Lewis, E. and A. Hoenes, "DNS Zone Transfer Protocol (AXFR)", RFC 5936, June 2010.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6644] Evans, D., Droms, R., and S. Jiang, "Rebind Capability in DHCPv6 Reconfigure Messages", RFC 6644, July 2012.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, October 2014.

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-dnssd-requirements]
Lynn, K., Cheshire, S., Blanchet, M., and D. Migault, "Requirements for Scalable DNS-SD/mDNS Extensions", draft-ietf-dnssd-requirements-06 (work in progress), March 2015.

[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-07 (work in progress), April 2015.

[RFC1033] Lottor, M., "Domain administrators operations guide", RFC 1033, November 1987.

[RFC4192] Baker, F., Lear, E., and R. Droms, "Procedures for Renumbering an IPv6 Network without a Flag Day", RFC 4192, September 2005.

[RFC7010] Liu, B., Jiang, S., Carpenter, B., Venaas, S., and W. George, "IPv6 Site Renumbering Gap Analysis", RFC 7010, September 2013.

[RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, September 2014.

[RFC7368] Chown, T., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", RFC 7368, October 2014.

Appendix A. Document Change Log

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

-07:

Ray Hunter is added as a co-author.

-06:

Ray Hunter is added in acknowledgment.

Adding Renumbering section with comments from Dallas meeting

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: January 6, 2016

M. Stenberg

S. Barth

P. Pfister
Cisco Systems
July 5, 2015

Home Networking Control Protocol
draft-ietf-homenet-hncp-07

Abstract

This document describes the Home Networking Control Protocol (HNCP), an extensible configuration protocol and a set of requirements for home network devices on top of the Distributed Node Consensus Protocol (DNCP). It enables automated configuration of addresses, naming, network borders and the seamless use of a routing protocol.

Table of Contents

1. Introduction

HNCP synchronizes state across a small site in order to allow automated network configuration. The protocol enables use of border discovery, address prefix distribution [I-D.ietf-homenet-prefix-assignment], naming and other services across multiple links.

HNCP provides enough information for a routing protocol to operate without homenet-specific extensions. In homenet environments where multiple IPv6 source-prefixes can be present, routing based on source and destination address is necessary [RFC7368].

2. 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

HNCP is defined as a profile of DNCP [I-D.ietf-homenet-dncp] with the following parameters:

- HNCP uses UDP datagrams on port HNCP-UDP-PORT as a transport over link-local scoped IPv6, using unicast and multicast (All-Homenet-Routers is the HNCP group address). Received datagrams with an IPv6 source or destination address which is not link-local scoped MUST be ignored. Unicast 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.

- HNCP operates on multicast-capable interfaces only. HNCP routers MUST assign a unique 32-bit endpoint identifier to each interface for which HNCP is enabled. The value zero is reserved for internal purposes. Implementations MAY use a value equivalent to the sin6_scope_id for the given interface.

- 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 DNCP Certificate Based Trust Consensus and MAY support the PKI-based trust method.

- HNCP uses opaque 32-bit node identifiers (DNCP_NODE_IDENTIFIER_LENGTH = 32). A node implementing HNCP SHOULD generate and use a random node identifier. If using a random node identifier and there is a node identifier collision, the node MUST immediately generate and use a new random node identifier which is not used by any other node.

- 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.

- HNCP nodes use the following Trickle parameters:

  * k SHOULD be 1, as the timer reset when data is updated and further retransmissions should handle packet loss.

  * 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 (i.e. 25.6 seconds) but MUST NOT be lower.

- HNCP nodes MUST use the leading 64 bits of MD5 [RFC1321] as DNCP non-cryptographic hash function H(x).

- HNCP nodes MUST use DNCP's keep-alive extension on all endpoints. The following parameters are suggested:

  * Default keep-alive interval (DNCP_KEEPALIVE_INTERVAL): 20 seconds.

  * Multiplier (DNCP_KEEPALIVE_MULTIPLIER): 2.1.

4. Common Links

HNCP uses the concept of Common Links for some of its applications. A Common Link usually refers to a link layer broadcast domain with certain properties and is used, e.g., to determine where prefixes should be assigned or which neighboring nodes participate in the election of a DHCP(v6) server. The Common Link is computed separately for each local interface, and it always contains the local

interface. Additionally, if the local interface is not in ad-hoc mode, it also contains the set of interfaces that are bidirectionally reachable from the given local interface, i.e. every remote interface of a remote node meeting all of the following requirements:

- The local node publishes a Neighbor TLV with:
  - Neighbor Node Identifier = remote node's node identifier
  - Neighbor Endpoint Identifier = remote interface's endpoint identifier
  - Endpoint Identifier = local interface's endpoint identifier
- The remote node publishes a Neighbor TLV with:
  - Neighbor Node Identifier = local node's node identifier
  - Neighbor 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 interfaces are connected, e.g. by detecting an identical remote interface being part of the Common Links of both local interfaces.

5. Border Discovery

HNCP router's interfaces are either internal, external or of a different category derived from the internal one. This section defines the border discovery algorithm. It is suitable for both IPv4 and IPv6 (single or dual-stack) and determines whether an HNCP interface is internal, external, or uses another fixed category. The algorithm is derived from the edge router interactions described in the Basic Requirements for IPv6 Customer Edge Routers [RFC7084]. This algorithm MUST be implemented by any router implementing HNCP.

The border discovery auto-detection algorithm works as follows, with evaluation stopping at first match:

1. If a fixed category is configured for the interface, it MUST be used.
2. If a delegated prefix could be acquired by running a DHCPv6 client on the interface, it MUST be considered external.

3. If an IPv4 address could be acquired by running a DHCPv4 client on the interface it MUST be considered external.

4. Otherwise the interface MUST be considered internal.

In order to avoid conflicts between border discovery and HNCP routers running DHCPv4 [RFC2131] or DHCPv6-PD [RFC3633] servers, each router MUST implement the following mechanism based on The User Class Option for DHCPv4 [RFC3004] and its DHCPv6 counterpart [RFC3315]:

- An HNCP router running a DHCP client on an HNCP interface MUST include a DHCP User-Class consisting of the ASCII-String "HOMENET".

- An HNCP router running a DHCP server on an HNCP interface MUST ignore or reject DHCP-Requests containing a DHCP User-Class consisting of the ASCII-String "HOMENET".

A router MUST allow setting a category of either auto-detected, internal or external for each interface which is suitable for both internal and external connections. In addition the following specializations of the internal category are defined to modify the local router behavior:

Leaf category: This declares an interface used by client devices only. Such an interface acts as an internal interface with the exception that HNCP or routing protocol traffic MUST NOT be sent on the interface, and all such traffic received on the interface MUST be ignored. This category SHOULD be supported.

Guest category: This declares an interface used by untrusted client devices only. In addition to the restrictions of the Leaf category, HNCP routers MUST enable firewalling rules 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.

Ad-hoc category: This configures an interface to be ad-hoc (Section 4). Ad-hoc interfaces are considered internal but no assumption is made on the the link transitivity properties. Support for this category is OPTIONAL.

Hybrid category: This declares an interface to be internal while still running DHCPv4 and DHCPv6-PD clients on it. It is assumed that the link is under control of a legacy, trustworthy non-HNCP router, still within 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.

Each router MUST continuously scan each active interface that does not have a fixed category in order to dynamically reclassify it if necessary. The router therefore runs an appropriately configured DHCPv4 and DHCPv6 client as long as the interface is active including states where it considers the interface to be internal. 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. Once it treats a certain interface as internal it MUST start forwarding traffic with appropriate source addresses between its internal interfaces and allow internal traffic to reach external networks according to the routes it publishes. Once a router detects an interface transitioning to external it MUST stop any previously enabled internal forwarding. In addition it SHOULD announce the acquired information for use in the network as described in later sections of this draft if the interface appears to be connected to an external network.

6. Autonomic Address Configuration

This section specifies how HNCP routers configure host and router addresses. At first border routers share information obtained from service providers or local configuration by publishing one or more External Connection TLVs. These contain other TLVs such as Delegated Prefix TLVs which are then used for prefix assignment. Finally, HNCP routers obtain addresses either statelessly or using a specific stateful mechanism and hosts and legacy routers are configured using SLAAC or DHCP.

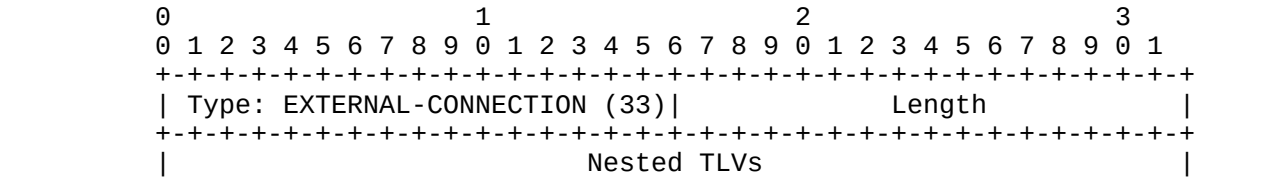
In all TLVs specified in this section which include a prefix, IPv4 prefixes are encoded using the IPv4-mapped IPv6 addresses format [RFC4291]. The prefix length of such IPv4 prefix is set to 96 plus the IPv4 prefix length.

6.1. External Connections

Each HNCP router MAY obtain external connection information from one or more sources, e.g., DHCPv6-PD [RFC3633], NETCONF [RFC6241] or static configuration. This section specifies how such information is encoded and advertised.

6.1.1. External Connection TLV

An External Connection TLV is a container-TLV used to gather network configuration information associated with a single external connection. A node MAY publish an arbitrary number of instances of this TLV.

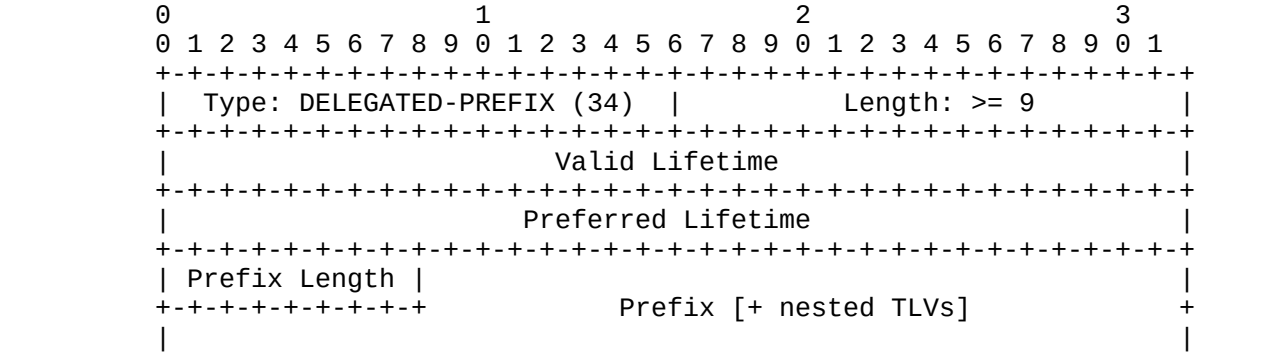


The External Connection TLV is a container which:

- MAY contain an arbitrary number of Delegated Prefix TLVs.
- MUST NOT contain multiple Delegated Prefix TLVs with identical or overlapping prefixes. In such a situation, the External Connection TLV MUST be ignored.
- MAY contain at most one DHCPv6 Data TLV and at most one DHCPv4 Data TLV encoding options associated with the External Connection but MUST NOT contain more than one of each otherwise the External Connection TLV MUST be ignored.
- MAY contain other TLVs for future use. Such additional TLVs MUST be ignored.

6.1.2. Delegated Prefix TLV

The Delegated Prefix TLV is used by HNCP routers to advertise prefixes which are allocated to the whole network and will be used for prefix assignment. Any Delegated Prefix TLV MUST be nested in an External Connection TLV.



Valid Lifetime: The time in seconds the delegated prefix is valid. The value is relative to the point in time the Node-Data TLV was last published. It MUST be updated whenever the node republishes its Node-Data TLV.

Preferred Lifetime: The time in seconds the delegated prefix is preferred. The value is relative to the point in time the Node-Data TLV was last published. It MUST be updated whenever the node republishes its Node-Data 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.

Nested TLVs: Other TLVs included in the Delegated Prefix TLV and starting at the next 32-bit boundary following the end of the encoded prefix:

* Zero or more Prefix Domain TLVs. In absence of any such TLV the prefix is assumed to be generated by an HNCP-router and for internal use only.

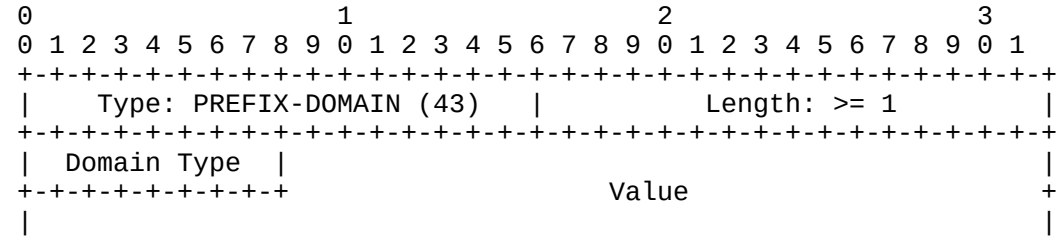
* If the encoded prefix represents an IPv6 prefix, at most one DHCPv6 Data TLV MAY be included, and any included DHCPv4 Data TLV MUST be ignored.

* If the prefix represents an IPv4 prefix (encoded as an IPv4-mapped IPv6 prefix), at most one DHCPv4 Data TLV MAY be included, and any included DHCPv6 Data TLV MUST be ignored.

* It MAY contain other TLVs for future use. Such additional TLVs MUST be ignored.

6.1.3. Prefix Domain TLV

The Prefix Domain TLV contains information about the origin and applicability of a delegated prefix. This information can be used to determine whether prefixes for a certain domain (e.g. local reachability, internet connectivity) do exist or should be acquired and to make decisions about assigning prefixes to certain links or to fine-tune border firewalls. See Section 6.5 for a more in-depth discussion.



Domain Type: The type of the domain identifier.

0 : Internet connectivity (no Value).

1-128 : Explicit destination prefix with the Domain Type being the actual length of the prefix (Value contains significant bits of the destination prefix padded with zeroes up to the next byte boundary).

129 : DNS Zone (Value contains an RFC 1035 [RFC1035] encoded DNS label sequence).

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

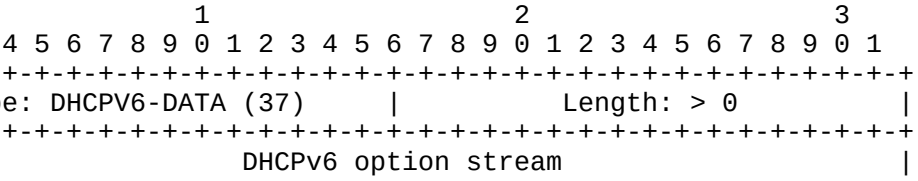
131-255: Reserved for future additions.

Value: A variable length identifier of the given type.

6.1.4. DHCP Data TLVs

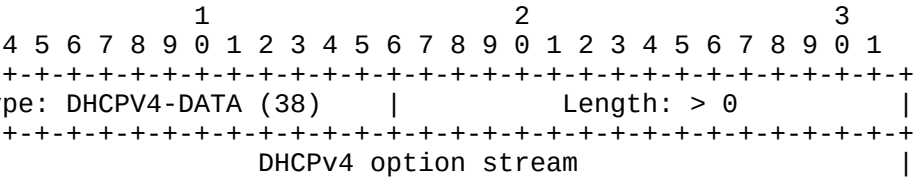
Auxiliary connectivity information is encoded as a stream of DHCP options. Such TLVs MUST only be present in an External Connection TLV or a Delegated Prefix TLV. When included in an External Connection TLV, they MUST contain DHCP options which are relevant to the whole External Connection. When included in a Delegated Prefix, they MUST contain DHCP options which are specific to the Delegated Prefix.

The DHCPv6 Data TLV uses the following format:



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

The DHCPv4 Data TLV uses the following format:



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

6.2. Prefix Assignment

HNCP uses the Distributed Prefix Assignment Algorithm specified in [I-D.ietf-homenet-prefix-assignment] in order to assign prefixes to HNCP internal links and uses the terminology defined there.

6.2.1. Prefix Assignment Algorithm Parameters

All HNCP nodes running the prefix assignment algorithm MUST use the following 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 internal, leaf, guest or ad-hoc interfaces. It is dynamically updated as HNCP 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 routers.

Set of Advertised Prefixes: The set of prefixes included in Assigned Prefix TLVs advertised by other HNCP routers (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 MAY be done instantly).

BACKOFF_MAX_DELAY: The default value is 4 seconds.

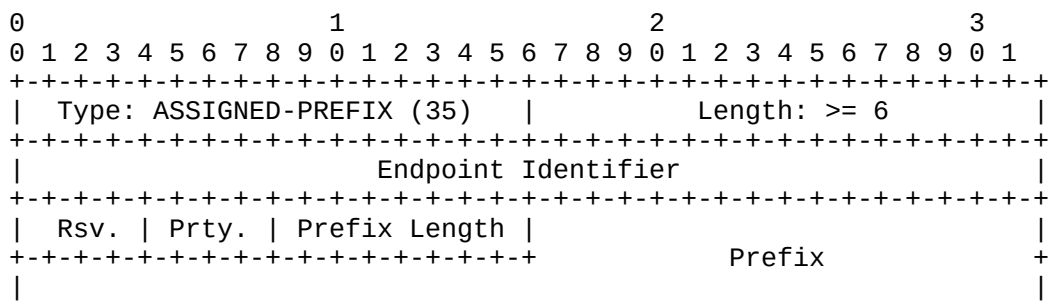
RANDOM_SET_SIZE: The default value is 64.

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.

6.2.2. Assigned Prefix TLV

Published Assigned Prefixes MUST be advertised using the Assigned Prefix TLV:



Endpoint Identifier: The endpoint identifier of the local interface that belongs to the Common Link the prefix is assigned to, or 0 if the Common Link is 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.

6.2.3. Making New Assignments

Whenever the Prefix Assignment Algorithm subroutine is run on a Common Link and whenever a new prefix may be assigned (case 1 of the subroutine), the decision of whether the assignment of a new prefix is desired MUST follow these rules:

If the Delegated Prefix TLV contains a DHCPv4 or DHCPv6 Data TLV, and the meaning of one of the DHCP options is not understood by the HNCP router, 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-null preferred lifetime, the creation of a new prefix is not desired.

Otherwise, the creation of a new prefix is desired, if the Delegated Prefix is either locally generated (does not have any Prefix Domain TLVs) or intended for internet access (has a Prefix

Domain TLV of type 0). Local desirability policies MAY override or provide additional desirability rules for delegated prefixes, e.g., by matching different Prefix Domain TLV values.

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 6.4 details how IPv4 delegated prefixes are generated), a prefix of length 24 SHOULD be preferred.

In any case, a router 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 support the L-capability and a router assigning an IPv6 prefix MUST support serving router advertisements. In addition if an assigned IPv6 prefix is not suitable for Stateless Address Autoconfiguration the router MUST also support the H-capability as defined in Section 10.

6.2.4. 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 create an appropriate route for said prefix, indicating it is directly reachable on the respective link and advertise said route using the chosen routing protocol.

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 6.3, e.g., if it is to be used as source for locally originating traffic.

6.2.5. DHCPv6-PD Excluded Prefix Support

Whenever a DHCPv6 Prefix Exclude option [RFC6603] is received with a delegated prefix, the excluded prefix MUST be advertised as assigned to a Private Link with the maximum priority (i.e. 15).

The same procedure MAY be applied in order to exclude prefixes obtained by other means of configuration.

6.2.6. Downstream Prefix Delegation Support

When an HNCP router receives a request for prefix delegation, it SHOULD assign one prefix per delegated prefix in the network. This set of assigned prefix is then delegated to the client, after it has been applied as described in the Prefix Assignment Algorithm. Each 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.

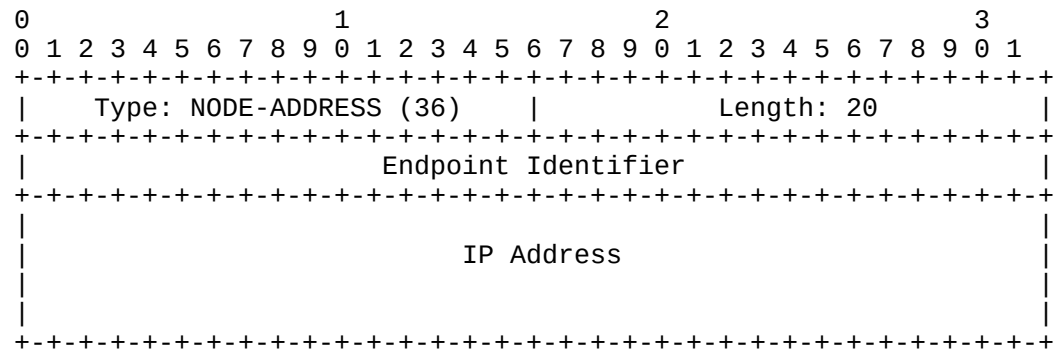
The assigned prefixes MUST NOT be given to clients before they are applied, and MUST be withdrawn whenever they are destroyed. As an exception to this rule, in order to shorten delays of processed requests, a router MAY prematurely give out a prefix which is advertised but not yet applied if it does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

6.3. 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 router MUST announce at least one IPv6 address and - if it supports IPv4 - at least one IPv4 address, whenever matching prefixes are assigned to at least one if 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 Modified EUI64 interface identifiers [RFC4291] SHOULD be used for address assignment whenever possible, otherwise (e.g., for IPv4) the following method MUST be used instead: For any assigned prefix for which SLAAC cannot be used, the first quarter of the addresses are reserved for routers HNCP based address assignments, whereas the last three quarters are left to the DHCPv6 (resp. DHCPv4) elected router (Section 10 specifies the DHCP server election process). For instance, 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 routers assign themselves addresses using the Node Address TLV:



Endpoint Identifier: The endpoint identifier of the local interface that belongs to the Common Link 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].

The process of obtaining addresses is specified as follows:

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

6.4. Local IPv4 and ULA Prefixes

HNCP routers can create an ULA or private IPv4 prefix to enable connectivity between local devices. These prefixes are inserted in HNCP as if they were delegated prefixes. 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

Domain 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 Domain of type 0 - if there is any. The router publishing a prefix with internet connectivity MUST announce an IPv4 default route using the routing protocol 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 other nodes 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].

6.5. Special Purpose Prefixes

Some 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.

Special purpose prefixes are distinguished using Prefix Domain TLVs (Section 6.1.3). 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 Domain TLVs other than type 0, as described in Section 6.2.3

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

The presence of a type 129 (DNS zone) Prefix Domain TLV indicates that the delegated prefix or its associated external connection is specialized to reach destinations within the given DNS zone. An HNCP router providing name resolving services SHOULD prefer DNS servers listed in the associated external connection's DHCPv4 or DHCPv6 Data TLVs when resolving domains from that zone.

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 10 in order to participate as a candidate in the election.

7.1. DHCPv6 for Addressing or Configuration

In general Stateless Address Autoconfiguration is used for client configuration for its low overhead and fast renumbering capabilities, 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 4) is elected based on the capabilities described in Section 10. The winner is the router (connected to the Common Link) advertising the greatest H-capability. In case of a tie, Capability Values and node identifiers are considered (greatest value is elected). The elected router MUST serve stateful DHCPv6 and MUST provide naming services for acquired hostnames as outlined in Section 8. 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. In case no router was elected, stateful DHCPv6 is not provided and each router assigning IPv6-prefixes on said link MUST provide stateless DHCPv6 service.

7.2. Sending Router Advertisements

All HNCP routers MUST send Router Advertisements periodically via multicast and via unicast in response to Router Solicitations.

- The "Managed address configuration" flag MUST be set whenever a router connected to the link is advertising a non-null H-capability and MUST NOT be set otherwise. The "Other configuration" flag MUST always be set.

- The default Router Lifetime MUST be set to an appropriate non-null value whenever an IPv6 default route is known in the HNCP network and MUST be set to zero otherwise.

- A Prefix Information Option MUST be added for each assigned and applied IPv6 prefix on the given link. The autonomous address-configuration flag MUST be set whenever the prefix is suitable for stateless configuration. The preferred and valid lifetimes MUST be smaller than the preferred and valid lifetimes of the delegated prefix the prefix is from. When a prefix is removed, it MUST be deprecated as specified in [RFC7084].

- A Route Information Option [RFC4191] MUST be added for each delegated IPv6 prefix known in the HNCP network. Additional ones SHOULD be added for each non-default IPv6 route with an external destination prefix advertised by the routing protocol.

- A Recursive DNS Server Option and a DNS Search List Option MUST be included with appropriate contents.

- To allow for optimized routing decisions for clients on the local link routers SHOULD adjust their Default Router Preference and Route Preferences [RFC4191] so that the priority is set to low if the next hop of the default or more specific route is on the same interface as the Route Advertisement being sent on. Similarly the router MAY use the high priority if it is certain it has the best metric of all routers on the link for all routes known in the network with the respective destination.

Every router sending Router Advertisements MUST immediately send an updated Router Advertisement via multicast as soon as it notices a condition resulting in a change of any advertised information.

7.3. DHCPv6 for Prefix Delegation

The designated DHCPv6 server for prefix-delegation on a Common Link is elected based on the capabilities described in Section 10. The winner is the router (connected to the Common Link) advertising the

greatest P-capability. In case of a tie, Capability Values are compared, and 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 6.2.6.

7.4. DHCPv4 for Addressing and Configuration

The designated DHCPv4 server on a Common Link (Section 4) is elected based on the capabilities described in Section 10. The winner is the router (connected to the Common Link) advertising the greatest L-capability. In case of a tie, Capability Values are compared, and 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.

The DHCPv4 serving router MUST announce itself as router [RFC2132] to clients if and only if there is an IPv4 default route known in the network. In addition, the router SHOULD announce a Classless Static Route Option [RFC3442] for each non-default IPv4 route advertised in the routing protocol with an external destination.

DHCPv4 lease times SHOULD be short (i.e. not longer than 5 minutes) in order to provide reasonable response times to changes.

7.5. Multicast DNS Proxy

The designated MDNS [RFC6762] proxy on a Common Link is elected based on the capabilities described in Section 10. The winner is the router (connected to the Common Link) advertising the greatest M-capability. In case of a tie, Capability Values are compared, and 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 announce an auto-generated or user-configured name for each internal Common Link (Section 4) for

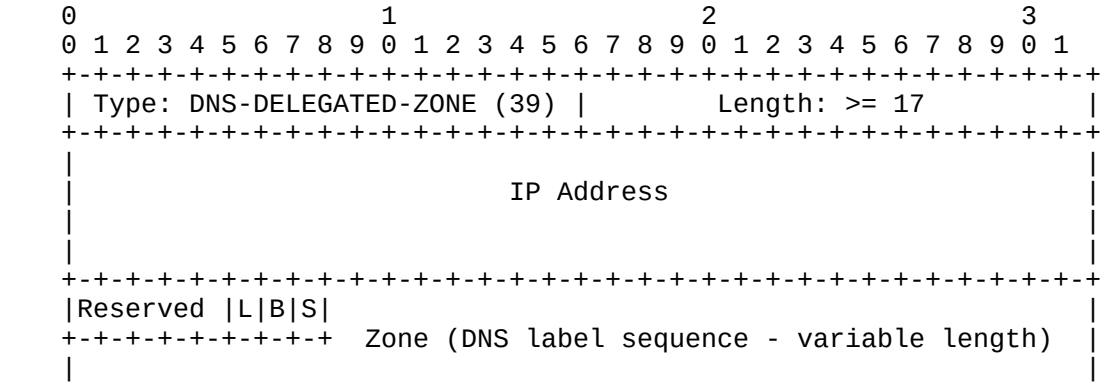
which it is the designated DHCPv4, stateful DHCPv6 server, MDNS proxy, or for which it provides forward or reverse DNS services on behalf of connected devices. HNCP routers providing name resolving services MUST use the included DNS server address to resolve names belonging to the zone.

Each HNCP router SHOULD announce a node name for itself to be easily reachable and MAY do so on behalf of other devices. HNCP routers providing name resolving services MUST resolve these names to their respective IP addresses.

The following TLVs are defined and MUST be supported by all nodes implementing naming and service discovery:

8.1. 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. There MUST NOT be more than one delegation for the same zone in the whole DNCP 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.



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.

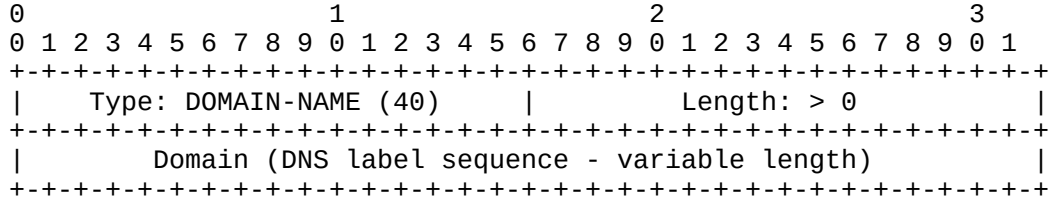
B-bit : (DNS-SD [RFC6763] Browse) indicates that this delegated zone should be included in the network's DNS-SD browse list of domains at b._dns-sd._udp. (DOMAIN-NAME). Local forward zones SHOULD have this bit set, reverse zones SHOULD NOT.

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 of the zone, encoded as the domain names are encoded DNS messages as specified in [RFC1035]. The last label in the zone MUST be empty.

8.2. Domain Name TLV

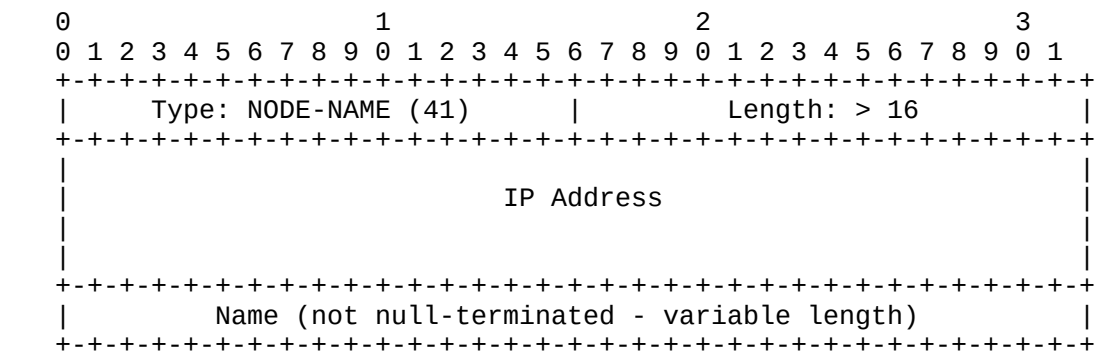
This TLV is used to indicate the base domain name for the network. It is the zone used as a base for all non fully-qualified delegated zones and node names. In case of conflicts the announced domain of the node with the greatest node identifier takes precedence. By default, i.e., if no node advertises such a TLV., ".home" is used. 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 zone MUST end with an empty label.

8.3. Node Name TLV

This TLV is used to assign the name of a node in the network to a certain IP address. In case of conflicts the announcement of the node with the greatest node identifier for a name takes precedence and all other nodes MUST cease to announce the conflicting TLV.



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

Name: The name of the node as a single DNS label (up to 63 characters, no leading length byte).

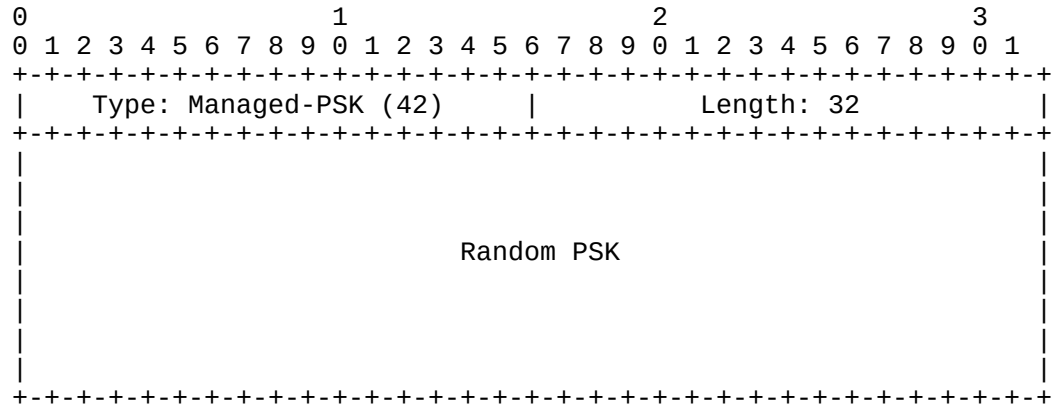
9. Securing Third-Party Protocols

Pre-shared keys (PSKs) are often required to secure 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 and SHOULD therefore be used if such a need arises. The following rules define how such a PSK is managed and used:

- If no Managed-PSK-TLV is currently being announced, an HNCP router 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.

- In case multiple routers 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.

- The router currently advertising the Managed-PSK-TLV must generate and advertise a new random one whenever an unreachable node is purged as described in DNCP.



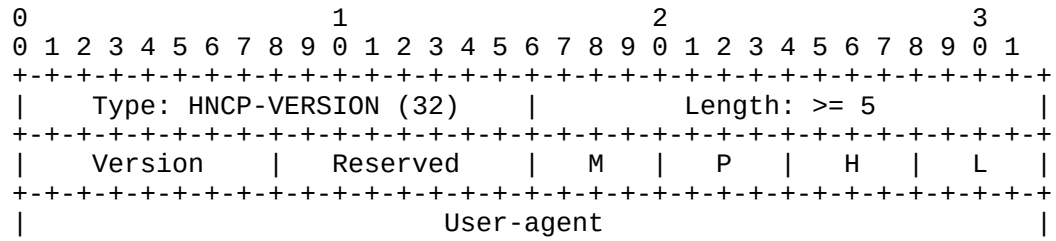
PSKs for individual protocols are derived from the random PSK through the use of HMAC-SHA256 [RFC6234] with a pre-defined per-protocol HMAC-key in ASCII-format. The following HMAC-keys are currently defined to derive PSKs for the respective protocols:

"ROUTING": to be used for IGPs

10. HNCP Versioning and Capabilities

Multiple versions of HNCP based on compatible DNCP profiles may be present in the same network when transitioning between HNCP versions and HNCP routers may have different capabilities to support clients. The following mechanism describes a way to announce the currently active version and User-agent of a node. Each node MUST include an HNCP-Version-TLV 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 or publishing such a TLV with a different Version number.

Capabilities are indicated by setting M, P, H and L fields in the TLV. The "capability value" is a metric indicated by interpreting the bits as an integer, i.e. (M << 12 | P << 8 | H << 4 | L).



Version: Indicates which version of HNCP is currently in use by this particular node. It MUST be set to 1. Nodes with different versions are considered incompatible.

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 some value between 1 and 7 included (4 is the default) if the router is capable of proxying MDNS and 0 otherwise. 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 some value between 1 and 7 included (4 is the default) if the router is capable of providing prefixes through DHCPv6-PD (Section 6.2.6) and 0 otherwise. 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 some value between 1 and 7 included (4 is the default) if the router is capable of providing such addresses and 0 otherwise. 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 some value between 1 and 7 included (4 is the default) if the router is capable of running a legacy DHCPv4 server offering IPv4 addresses to clients and 0 otherwise. 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.

11. Requirements for HNCP Routers

Each router implementing HNCP is subject to the following requirements:

- It MUST implement HNCP-Versioning, Border Discovery, Prefix Assignment and Configuration of hosts and non-HNCP routers as defined in this document.

- It MUST implement and run the method for securing third-party protocols whenever it uses the security mechanism of HNCP.

- It SHOULD implement support for the Service Discovery and Naming TLVs as defined in this document.

- It MUST implement and run a routing protocol appropriate for the given link type on all of the interfaces it sends and receives HNCP traffic on. The protocol MUST support source-specific routes and MUST correctly propagate those also for the external destinations that may have only implicit source-specific information, such as a combination of a DHCPv6 PD-derived prefix and a non-source-specific default route.

- It MUST use adequate security mechanisms for the routing protocol on any interface where it also uses the security mechanisms of HNCP. If the security mechanism is based on a PSK it MUST use a PSK derived from the Managed-PSK to secure the IGP.

- It MAY be able to provide connectivity to IPv4-devices using DHCPv4.

- It SHOULD be able to delegate prefixes to legacy IPv6 routers using DHCPv6-PD.

- In addition, normative language of Basic Requirements for IPv6 Customer Edge Routers [RFC7084] applies with the following adjustments:

  * The section "WAN-Side Configuration" applies to HNCP 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 HNCP interfaces classified as internal.

  * The requirement L-2 to assign a separate /64 to each LAN interface is replaced by the participation in the prefix assignment mechanism (Section 6.2) for each such interface.

* 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.

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:

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

12.1. Border Determination

As described in Section 5, an HNCP router determines the internal or external state on a per-link basis. A firewall perimeter is set up for the external links, and for internal links, HNCP and IGP traffic is allowed.

Threats concerning automatic border discovery 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 adequate 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 eavesdropping, spoofing or denial of service attacks on HNCP services such as address assignment or service discovery.

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 and IGP 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, DNCP secure mode 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 is requested to maintain a registry for HNCP TLV-Types. This registry inherits TLV-Types and allocation policy defined in DNCP [I-D.ietf-homenet-dncp], but is independent with regard to all TLV-Types not specified or reserved by DNCP. Particularly, other DNCP profile may have there own registries, using same TLV numbers.

The following TLV-Types are defined in this document:

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

HNCP requires allocation of UDP port numbers HNCP-UDP-PORT and HNCP-DTLS-PORT, as well as an IPv6 link-local multicast address All-Homenet-Routers.

14. References

14.1. Normative references

[I-D.ietf-homenet-dncp]
Stenberg, M. and S. Barth, "Distributed Node Consensus Protocol", draft-ietf-homenet-dncp-07 (work in progress), July 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6603] Korhonen, J., Savolainen, T., Krishnan, S., and O. Troan, "Prefix Exclude Option for DHCPv6-based Prefix Delegation", RFC 6603, May 2012.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

[I-D.ietf-homenet-prefix-assignment] Pfister, P., Paterson, B., and J. Arkko, "Distributed Prefix Assignment Algorithm", draft-ietf-homenet-prefix-assignment-07 (work in progress), June 2015.

14.2. Informative references

[RFC7084] Singh, H., Beebee, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, November 2013.

[RFC3004] Stump, G., Droms, R., Gu, Y., Vyaghrapuri, R., Demirtjis, A., Beser, B., and J. Privat, "The User Class Option for DHCP", RFC 3004, November 2000.

[RFC3118] Droms, R. and W. Arbaugh, "Authentication for DHCP Messages", RFC 3118, June 2001.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

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

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

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC7368] Chown, T., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", RFC 7368, October 2014.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, March 1997.

[RFC3442] Lemon, T., Cheshire, S., and B. Volz, "The Classless Static Route Option for Dynamic Host Configuration Protocol (DHCP) version 4", RFC 3442, December 2002.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

14.3. URIs

[3] http://www.openwrt.org

[4] http://www.homewrt.org/doku.php?id=run-conf

Appendix A. Changelog [RFC Editor: please remove]

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 [3] package repositories. See [4] for more information. Feedback and contributions are welcome.

Appendix D. Acknowledgements

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

Thanks to Eric Kline for the original border discovery work.

Authors' Addresses

Markus Stenberg
Helsinki 00930
Finland

Email: markus.stenberg@iki.fi

Steven Barth
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: September 6, 2015

M. Stenberg

March 5, 2015

Auto-Configuration of a Network of Hybrid Unicast/Multicast DNS-Based Service Discovery Proxy Nodes
draft-ietf-homenet-hybrid-proxy-zeroconf-00

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 September 6, 2015.

Table of Contents

1. Introduction

Section 3 ("Hybrid Proxy Operation") of [I-D.cheshire-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.cheshire-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:

- 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).

- 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).

- 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.

- 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.

- 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. 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).

7. IANA Considerations

This document has no actions for IANA.

8. References

8.1. Normative references

[I-D.cheshire-dnssd-hybrid]
Cheshire, S., "Hybrid Unicast/Multicast DNS-Based Service Discovery", draft-cheshire-dnssd-hybrid-01 (work in progress), January 2014.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

8.2. Informative references

[I-D.ietf-homenet-hncp]
Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", draft-ietf-homenet-hncp-03 (work in progress), January 2015.

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

[RFC3646] Droms, R., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, December 2003.

8.3. 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.

Author’s Address

Markus Stenberg
Helsinki 00930
Finland

Email: markus.stenberg@iki.fi

HOMENET
Internet-Draft
Intended status: Standards Track
Expires: November 20, 2015

D. Migault (Ed)
Ericsson
W. Cloetens
SoftAtHome
C. Griffiths
Dyn
R. Weber
Nominum

May 19, 2015

DHCP Options for Homenet Naming Architecture
draft-ietf-homenet-naming-architecture-dhc-options-02.txt

Abstract

CPEs are usually constraint 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. One way to avoid exposing CPE is to outsource the authoritative service to a third party. 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 which may be unappropriated for most end users. To leverage this issue, this document proposes DHCP Options so any agnostic CPE can automatically proceed to the appropriated configuration and outsource the authoritative naming service for the home network. This document shows that in most cases, these DHCP Options make outsourcing to a third party (be it the ISP or any ISP independent service provider) transparent for the end user.

This Internet-Draft will expire on November 20, 2015.

Copyright Notice

Table of Contents

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

- Customer Premises Equipment: (CPE) is the router providing connectivity to the home network. It is configured and managed by the end user. In this document, the CPE might also hosts services such as DHCPv6. This device might be provided by the ISP.

- Public Key: designates a public Key generated by the CPE. This key is used as an authentication credential for the CPE.

- Registered Homenet Domain: is the Domain Name associated to the home network.

- DNS Homenet Zone: is the DNS zone associated to the home network. This zone is set by the CPE and essentially contains the bindings between names and IP addresses of the nodes of the home network. In this document, the CPE does neither perform any DNSSEC management operations such as zone signing nor provide an authoritative service for the zone. Both are delegated to the Public Authoritative Server. The CPE synchronizes the DNS Homenet Zone with the Public Authoritative Server via a hidden primary / secondary architecture. The Public Authoritative Server might use specific servers for the synchronization of the DNS Homenet Zone: the Public Authoritative Name Server Set.

- DNS Homenet Zone Template: The template used as a basis to generate the DNS Homenet Zone.

- DNS Template Server: The DNS server that hosts the DNS Homenet Zone Template.

- DNS Homenet Reverse Zone: The reverse zone file associated to the DNS Homenet Zone.

- Public Authoritative Primary(ies): are the visible name server hosting the DNS Homenet Zone. End users' resolutions for the Homenet Domain are sent to this server, and this server is a primary for the zone.

- Public Authoritative Name Server Set: is the server the CPE synchronizes the DNS Homenet Zone. It is configured as a secondary and the CPE acts as primary. The CPE sends information so the DNSSEC zone can be set and served.

- Reverse Public Authoritative Primary(ies): are the visible name server hosting the DNS Homenet Reverse Zone. End users' resolutions for the Homenet Domain are sent to this server, and this server is a primary for the zone.

- Reverse Public Authoritative Name Server Set: is the server the CPE synchronizes the DNS Homenet Reverse Zone. It is configured as a secondary and the CPE acts as primary. The CPE sends information so the DNSSEC zone can be set and served.

3. Introduction

CPEs are usually constraint 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. One way to avoid exposing CPE is to outsource the authoritative service to a third party. 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 which may be unappropriated for most end users. To leverage this issue, this document proposes DHCP Options so any agnostic CPE can automatically proceed to the appropriated configuration and outsource the authoritative naming service for the home network. This document shows that in most cases, these DHCP Options make outsourcing to a third party (be it the ISP or any ISP independent service provider) transparent for the end user.

When the CPE is plugged, the DHCP Options described in the document enable the CPE:

- 1. To build the DNS Homenet Zone: Building the DNS Homenet Zone requires filling the zone with appropriated bindings likes name / IP addresses of the different devices in the home networks. Such information can be provided for example by the DHCP Server hosted on the CPE. On the other hand, it also requires configuration parameters like the name of the Registered Domain Name associated to the home network or the Public Authoritative Primary(ies) the DNS Homenet Zone is outsourced to. These configuration parameters are stored in the DNS Homenet Zone Template. This document describes the DHCP Zone Template Option. This option carries a DNS Homenet Zone Template FQDN. In order to retrieve the DNS Homenet Zone Template, the CPE sends a query of type AXFR [RFC1034] [RFC5936]for the DNS Homenet Zone Template FQDN.

- 2. To upload the DNS(SEC) Homenet Zone to the appropriated server: This server is designated as the Public Authoritative Name Server Set. It is in charge of publishing the DNS(SEC) Homenet Zone on the Public Authoritative Primary(ies). This document describes the DHCP Public Authoritative Name Server Set Option that provides the FQDN of the appropriated server. Note that, in the document we do not consider whether the DNS(SEC) Homenet Zone is signed or not and if signed who signs it. Such questions are out of the scope of the current document.

- 3. To upload the DNS Homenet Reverse Zone to the appropriated server: This server is designated as the Reverse Public Authoritative Name Server Set. It is in charge of publishing the DNS Homenet Reverse Zone on the Reverse Public Authoritative Primary(ies). This document describes the DHCP Reverse Public Authoritative Name Server Set Option that provides the FQDN of the appropriated server. Similarly to item 2., we do not consider in this document if the DNS 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 DNS Homenet Zone Template, the DNS(SEC) Homenet Zone and DNS 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 DHCP Public Key Option. This option is sent by the CPE to the DHCP Server and provides the Public Key the CPE uses to authenticate itself. The DHCP Server is then responsible to provide the Public Key to the various DNS servers.

As a result, the DHCP Options described in this document enable an agnostic CPE to outsource its naming infrastructure without any configuration from the end user. The main reason no configuration is required by the end user is that there are privileged links: first between the CPE and the DHCP Server and then between the DHCP Server and the various DNS servers (DNS Homenet Zone Server, the Reverse Public Authoritative Name Server Set, Public Authoritative Name Server Set). This enables the CPE to send its authentication credentials (a Public Key) to the DHCP Server that in turn forward it to the various DNS servers. With the authentication credential on the DNS servers set, the CPE is able to update the various zones in a secure way.

If the DHCP Server cannot provide the public key to one of these servers (most likely the Public Authoritative Name Server Set) and the CPE needs to interact with the server, then, the end user is expected to provide the CPE's public key to these servers (the Reverse Public Authoritative Name Server Set or the Public Authoritative Name Server Set) 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 as follows. Section 4 provides an overview of the DHCP 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 DHCP Options and Section 7 details how DHCP Client DHCP Server and DHCP Relay 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 how the CPE is expect to interact with various entities, as well as how the CPE is expected to be configured via DHCP Options. Section 4.1 describes the entities the CPE is expected to interact with. Interaction with each entities is defined via DHCP Options that are expected to configure the CPE. Once configured, the CPE is expected to be able to update some DNS Data hosted by the different entities. As a result security and updating mechanisms play an important role in the specification. Section 4.2 provides an overview of the different security mechanisms considered for securing the CPE transactions and Section 4.3

considers the different update mechanisms considered for the CPE to update the DNS Data.

4.1. Architecture and DHCP Options Overview

This section illustrates how a CPE configures its naming infrastructure to outsource its authoritative naming service. All configurations and settings are performed using DHCP Options. This section, for the sake of simplicity, assumes that the DHCP Server is able to communicate to the various DNS servers and to provide them the public key associated to the CPE. Once each server got the public key, the CPE can proceed to updates in a 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 Public Authoritative Name Server Set 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 Public Key to the DHCP Server using a DHCP Public Key Option (OPTION_PUBLIC_KEY) and sends a DHCP Option Request Option (ORO) for the DHCP Zone Template Option (OPTION_DNS_ZONE_TEMPLATE), the DHCP Public Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET) and the DHCP Reverse Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET).

- 2: The DHCP Server makes the Public Key available to the DNS servers, so the CPE can secure its DNS transactions. Note that the 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 were the DHCP Server forwards the set (Public Key, Zone Template FQDN) to the DNS Template Server, the set (Public Key, IPv6 subnet) to the Reverse Public Authoritative Name Server Set and the set (Public Key, Registered Homenet Domain) to the Public Authoritative Name Server Set.

- 3.: The DHCP Server responds to the CPE with the requested DHCP Options, i.e. the DHCP Public Key Option (OPTION_PUBLIC_KEY), DHCP Zone Template Option OPTION_DNS_ZONE_TEMPLATE, DHCP Public

Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET), DHCP Reverse Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET).

- 4.: Upon receiving the DHCP Zone Template Option (OPTION_DNS_ZONE_TEMPLATE), the CPE performs an AXFR DNS query for the Zone Template FQDN. The exchange is secured according to the security protocols defined in the Security field of the DHCP option. Once the CPE has retrieved the DNS Zone Template, the CPE can build the DNS Homenet Zone and the DNS Homenet Reverse Zone. Eventually the CPE signs these zones.

- 5.: Once the DNS(SEC) Homenet Reverse Zone has been set, the CPE uploads the zone to the Reverse Public Authoritative Name Server Set. The DHCP Reverse Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET) provides the Reverse Public Authoritative Name Server Set FQDN as well as the upload method, and the security protocol to secure the upload.

- 6.: Once the DNS(SEC) Homenet Zone has been set, the CPE uploads the zone to the Public Authoritative Name Server Set. The DHCP Public Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET) provides the Public Authoritative Name Server Set FQDN as well as the upload method and the security protocol to secure the upload.

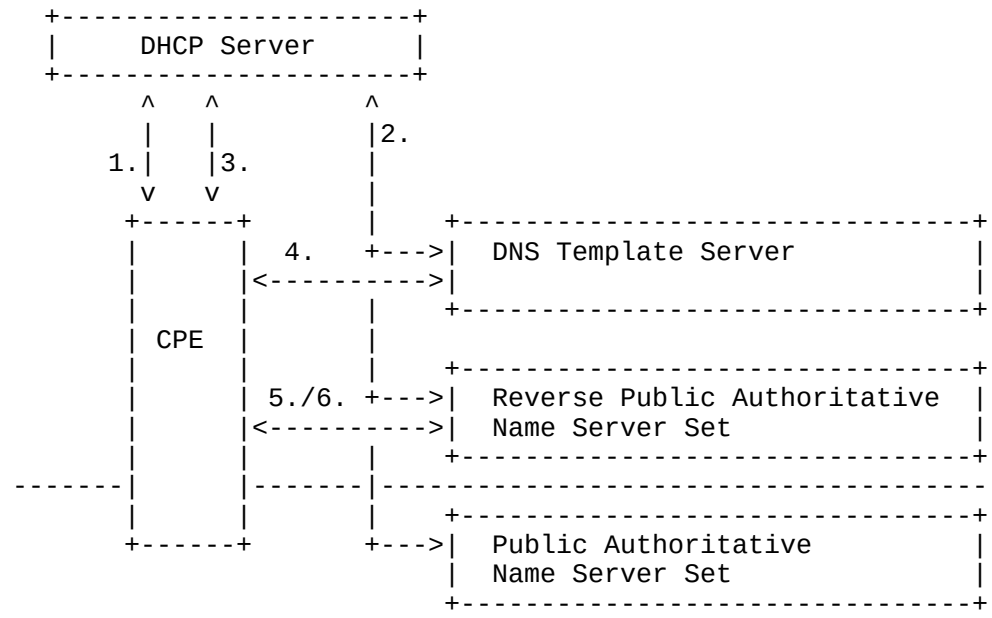


Figure 1: Protocol Overview

As described above, the CPE is likely to interact with various DNS content. This section is focused on DNS Data the CPE is likely to update. More specifically, the CPE is likely to update the:

- DNS Homenet Zone Template: may be updated by the CPE if the configuration of the zone may be changed. This can include additional Public Authoritative Primary(ies), a different Registered Homenet Domain as the one initially proposed, or a redirection to another domain.

- DNS Homenet Reverse Zone: may be updated every time a new device is connected or dis-connected.

- DNS Homenet Zone: may be updated every time a new device is connected, dis-connected.

In fact, the CPE must be able to perform these updates in a secure manner. There are multiple ways to secure a DNS transaction and this document considers two mechanisms to update a DNS Data (nsupdate and primary/secondary synchronization). Which security mechanism to use to secure a DNS transaction depends on the expected security (authentication of the authoritative server, mutual authentication, confidentiality...). The expected security may also depends on the kind of transaction performed by the CPE. Section 4.2 describes the

different security mechanisms considered in the document as well as their respective goals. Which mechanism to use to update the DNS Data depends on the kind of update. Frequency of the update, size of the DNS Data to update may be some useful criteria. Section 4.3 positions the nsupdate and primary/secondary synchronization mechanisms.

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 restricts the scope of security protocols to those 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 provides means to distribute shared secret for example using a specific DHCP 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).

Exchange with the DNS Template Server to retrieve the DNS 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 prevents someone else to query the DNS 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 DNS Homenet Zone. In fact, these DNSSEC related fields

are associated to the DNS Homenet Zone Template and not the DNS 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 DNS Homenet Zone Template can only be updated with DNS update. The reason is that the DNS Homenet Zone Template contains static configuration data that is not expected to evolve over time.

The DNS Homenet Reverse Zone and the DNS 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 then a NOTIFY, and they do not provide the server the opportunity to post-pone 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 is configured as a primary whereas the DNS Server is configured as a secondary. The DNS Server represents the Public Authoritative Name Server Set or the Reverse Public Authoritative Name Server Set.

When the CPE is plugged its IP address may be unknown to the secondary. The section details how the CPE or primary communicate 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 / Public Authoritative Name Server Set

The CPE knows the zone to be synchronized by reading the Registered Homenet Domain in the DNS Homenet Zone Template provided by the DHCP Zone Template Option (OPTION_DNS_ZONE_TEMPLATE). The IP address of the secondary is provided by the DHCP Public Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET).

The Public Authoritative Name Server Set has been configured with the Registered Homenet Domain and the Public Key that identifies the CPE. The only thing missing 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 DNS Homenet Zone, it sends a NOTIFY message to the Public Authoritative Name Server Sets. 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 Public Authoritative Name Server Sets might consider the source IP address of the NOTIFY query to configure the primaries attributes.

5.1.2. CPE / Reverse Public Authoritative Name Server Set

The CPE knows the zone to be synchronized by looking at its assigned prefix. The IP address of the secondary is provided by the DHCP Reverse Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET).

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. DNS Homenet Zone Template

The DNS Homenet Zone Template contains at least the related fields of the Public Authoritative Primary(ies) 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 Primary(ies). This default settings should provide the CPE the necessary pieces of information to set the homenet naming architecture.

If the DNS Homenet Zone Template is not subject to modifications or updates, the owner of the template might only use DNSSEC to enable integrity check.

The DNS Homenet Zone Template might 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 DNS Homenet Zone Template. Policies that defines what is accepted or rejected is out of scope of this document. However, in this document we assume the Registered Homenet Domain is used as an index by the Public Authoritative Name Server Set, and SIG(0), TSIG are used to authenticate the CPE. As a result, the Registered Homenet Domain should not be modified unless the Public Authoritative Name Server Set can handle with it.

5.2.2. DNS (Reverse) Homenet Zone

The DNS Homenet Zone might be generated from the DNS Homenet Zone Template. How the DNS Homenet Zone is generated is out of scope of this document. In some cases, the DNS Homenet Zone might be the exact copy of the DNS Homenet Zone Template. In other cases, it might be generated from the DNS Homenet Zone Template with additional RRsets. In some other cases, the DNS Homenet Zone might be generated without considering the DNS 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 DNS Homenet Zone Template. This constrain 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 DNS 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 DHCP Options. A few DHCP Options are used to advertise a server the CPE may be expect to interact with. Interaction may require to define how the update is expected to be performed as well as how the communication is secured. Security and Update are shared by multiple DHCP Options and are described in separate sections. Section 6.1 describes the security field, Section 6.2 describes the update fields, the remaining sections Section 6.3, Section 6.4, Section 6.5, Section 6.6 describe the DHCP Options.

6.1. Security Field

The Security Field of the DHCP Option is represented in Figure 2. It indicates the security mechanism 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.

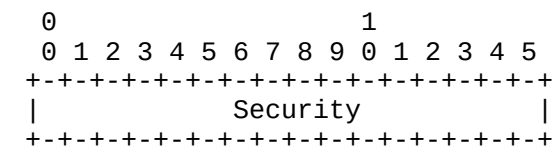


Figure 2: Security Field

- 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 DNS 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 ignored by the DHCP Client.

A Security field with all bits set to zero indicates the operation is not permitted. The Security 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 DHCP Option is represented in Figure 3. It indicates the update mechanism supported by the DNS server. See Section 4.3 for more details.

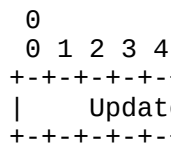


Figure 3: Update Field

- 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 DHCP Server and ignored by the DHCP Client.

6.3. DHCP Public Key Option

The DHCP Public Key Option (OPTION_PUBLIC_KEY) indicates the Public Key that is used to authenticate the CPE.

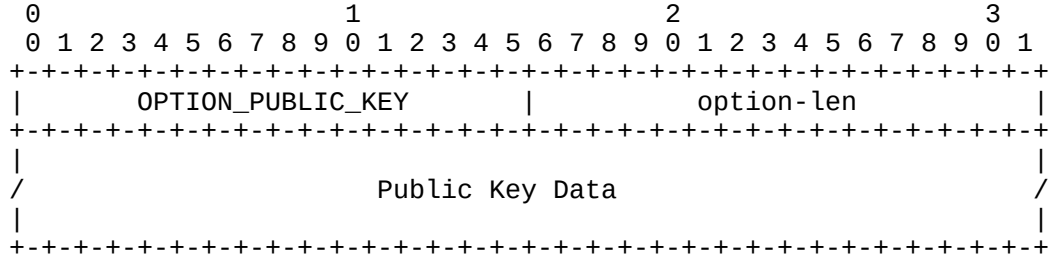


Figure 4: DHCP Public Key Option

- OPTION_PUBLIC_KEY (variable): the option code for the DHCP Public Key Option.

- option-len (16 bits): length in octets of the option-data field as described in [RFC3315].

- Public Key Data: contains the Public Key. The format is the DNSKEY RDATA format as defined in [RFC4034].

6.4. DHCP Zone Template Option

The DHCP Zone Template Option (OPTION_DNS_ZONE_TEMPLATE) Option indicates the CPE how to retrieve the DNS Homenet Zone Template. It provides a FQDN the CPE SHOULD query with a DNS query of type AXFR. The option also specifies which security protocols are available on the authoritative server. DNS Homenet Zone Template update, if permitted MUST use the DNS Update mechanism.

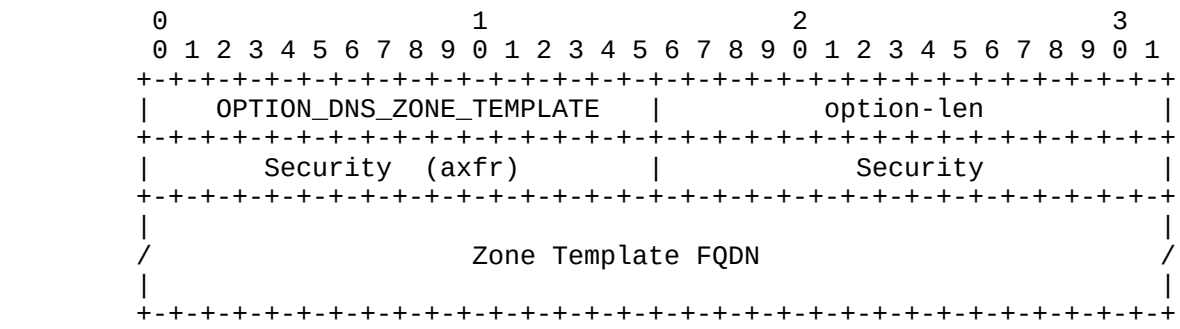


Figure 5: DHCP Zone Template Option

- OPTION_DNS_ZONE_TEMPLATE (variable): the option code for the DHCP Zone Template Option.

- option-len (16 bits): length in octets of the option-data field as described in [RFC3315].

- Security (axfr) (16 bits): defines which security protocols 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.

- Security (16 bits): defines which security protocols 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 DNS Homenet Zone Template.

6.5. DHCP Public Authoritative Name Server Set Option

The DHCP Public Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET) provides information so the CPE can upload the DNS Homenet Zone to the Public Authoritative Name Server Set. Finally, the option provides the security mechanisms that are

available to perform the upload. The upload is performed via a DNS primary / secondary architecture or DNS updates.

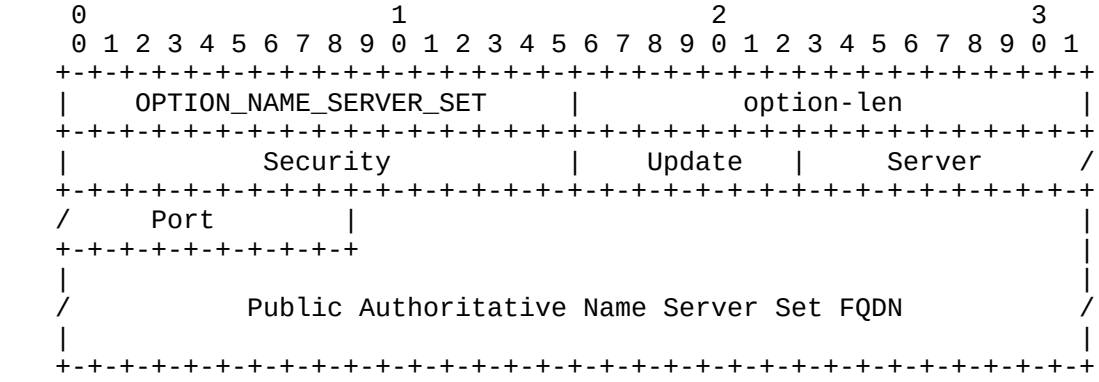


Figure 6: DHCP Public Authoritative Name Server Set Option

- OPTION_NAME_SERVER_SET (16 bits): the option code for the DHCP Public Authoritative Name Server Set Option.

- option-len (16 bits): length in octets of the option-data field as described in [RFC3315].

- Security (16 bits): defines which security protocols 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 Public Authoritative Name Server Set is listening.

- Public Authoritative Name Server Set FQDN (variable): the FQDN of the Public Authoritative Name Server Set.

6.6. DHCP Reverse Public Authoritative Name Server Set Option

The DHCP Reverse Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET) provides information so the CPE can upload the DNS Homenet Zone to the Public Authoritative Name Server Set. The option provides the security mechanisms that are available to perform the upload. The upload is performed via a DNS primary / secondary architecture or DNS updates.

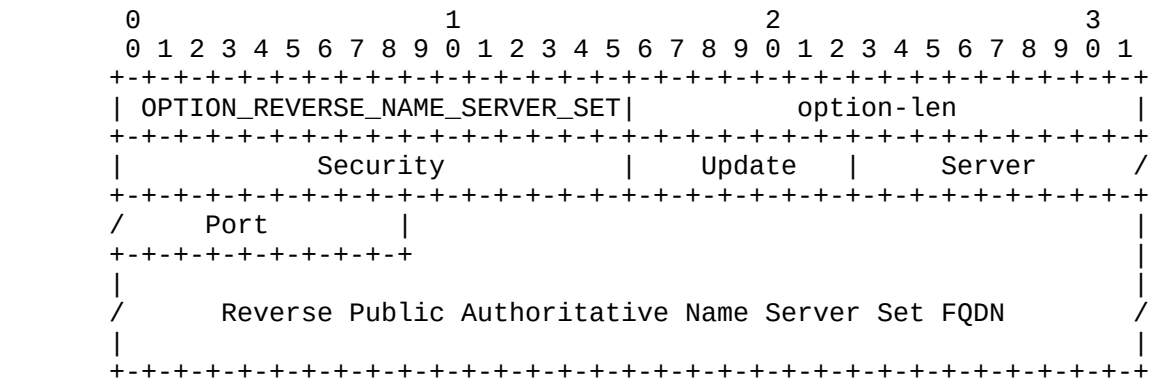


Figure 7: DHCP Reverse Public Authoritative Name Server Set Option

- OPTION_REVERSE_NAME_SERVER_SET (16 bits): the option code for the DHCP Reverse Public Authoritative Name Server Set Option.

- option-len (16 bits): length in octets of the option-data field as described in [RFC3315].

- Security (16 bits): defines which security protocols 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 Public Authoritative Name Server Set is listening.

- Reverse Public Authoritative Name Server Set FQDN (variable): The FQDN of the Reverse Public Authoritative Name Server Set.

7. DHCP Behavior

7.1. DHCPv6 Server Behavior

The DHCP Server sends the DHCP Zone Template Option (OPTION_DNS_ZONE_TEMPLATE), DHCP Public Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET), DHCP Reverse Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET) upon request by the DHCP Client.

The DHCP Server MAY receive a DHCP Public Key Option (OPTION_PUBLIC_KEY) from the CPE. Upon receipt of this DHCP Option, the DHCP Sever is expect to communicate this credential to the available DNS Servers like the DNS Template Server, the Public

Authoritative Name Server Set and the Reverse Public Authoritative Name Server Set.

7.2. DHCPv6 Client Behavior

The DHCP Client MAY send a DHCP Public Key Option (OPTION_PUBLIC_KEY) to the DHCP Server. This Public Key authenticates the CPE.

The DHCP Client sends a DHCP Option Request Option (ORO) with the necessary DHCP options.

A CPE SHOULD only send the an ORO request for DHCP Options it needs or for information that needs to be up-to-date.

Upon receiving a DHCP option described in this document, the CPE SHOULD retrieve or update DNS zones using the associated security and update protocols.

7.3. DHCPv6 Relay Behavior

DHCP Relay behavior are not modified by this document.

8. IANA Considerations

The DHCP options detailed in this document is:

- OPTION_DNS_ZONE_TEMPLATE: TBD

- OPTION_NAME_SERVER_SET: TBD

- OPTION_REVERSE_NAME_SERVER_SET: TBD

- OPTION_PUBLIC_KEY: TBD

9. Security Considerations

9.1. DNSSEC is recommended to authenticate DNS hosted data

It is recommended that the (Reverse) DNS 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 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.

In fact, the channel MUST be secured because the CPE provides authentication credentials. Unsecured channel may result in CPE impersonation attacks.

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. Acknowledgment

We would like to thank Tomasz Mrugalski, Marcin Siodelski and Bernie Volz for their comments on the design of the DHCP 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.

11. References

11.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, August 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, May 2000.

[RFC2930] Eastlake, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, September 2000.

[RFC2931] Eastlake, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, September 2000.

[RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, November 2000.

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

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5936] Lewis, E. and A. Hoenes, "DNS Zone Transfer Protocol (AXFR)", RFC 5936, June 2010.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, June 2012.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, October 2014.

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-homenet-front-end-naming-delegation]
Migault, D., Cloetens, W., Griffiths, C., and R. Weber, "Outsourcing Home Network Authoritative Naming Service", draft-ietf-homenet-front-end-naming-delegation-02 (work in progress), May 2015.

[I-D.sury-dnsext-cname-dname]
Sury, O., "CNAME+DNAME Name Redirection", draft-sury-dnsext-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 Primary(ies) the ISP is able to build the DNS Homenet Zone Template.

The ISP manages the DNS Template Server, so it is able to load the DNS Homenet Zone Template on the DNS Template Server.

When the CPE is plugged (at least the first time), it provides its 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:

- Public Authoritative Name Server Set

- Reverse Public Authoritative Name Server Set

- 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-dnsext-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 Public Authoritative Name Server Set and Public Authoritative Primary(ies) to a third party. All other DNS Servers DNS Template Server, Reverse Public Authoritative Primary(ies) and Reverse Public Authoritative Name Server Set remain managed by the ISP. The reason we consider that Reverse Public Authoritative Primary(ies) and Reverse Public Authoritative Name Server Set 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 Public Authoritative Name Server Set and Public Authoritative Primary(ies) to a third party. Similarly, the DNS Template Server can be easily outsourced as detailed in this section

Outsourcing Public Authoritative Name Server Set and Public Authoritative Primary(ies) requires:

- 1) Updating the DNS 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 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 Primary(ies), with a button "Configure DNS Homenet Zone Template".

- 2) Updating the DHCP Server Information. In fact the Reverse Public Authoritative Name Server Set 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 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 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 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 DHCP Public Key Option as well as a request for a DHCP Zone Template Option, a DHCP Public Authoritative Name Server Set Option and a DHCP Reverse Public Authoritative Name Server Set 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 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 DHCP 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 DHCP 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 Public Authoritative Name Server Set 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 DHCP 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 security mechanisms have been intergrated 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 DHCP 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 DHCP 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 DHCP 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

Wouter Cloetens
SoftAtHome
vaartdijk 3 701
3018 Wijgmaal
Belgium

Email: wouter.cloetens@softathome.com

Chris Griffiths
Dyn
150 Dow Street
Manchester, NH 03101
US

Email: cgriffiths@dyn.com
URI: http://dyn.com

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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 17, 2015

P. Pfister
B. Paterson
Cisco Systems
J. Arkko
Ericsson

June 15, 2015

Distributed Prefix Assignment Algorithm
draft-ietf-homenet-prefix-assignment-07

Abstract

This document specifies a distributed algorithm for automatic prefix assignment. Thus it provides an alternative to manual or centralized prefix and address assignment techniques. Given a set of delegated prefixes, it ensures that at most one prefix is assigned from each delegated prefix to each link. Nodes may assign available prefixes to the links they are directly connected to, or for other private purposes. The algorithm eventually converges and ensures that all assigned prefixes do not overlap.

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 December 17, 2015.

Copyright Notice

Table of Contents

1. Introduction

This document specifies a distributed algorithm for automatic prefix assignment. The algorithm provides a generic alternative to centralized (human or software based) approaches for network prefixes and addresses assignment. Although it does not require to be configured to operate properly, it supports custom configuration by means of variable priority assignments, and can therefore be used in fully autonomic as well as professionally managed networks.

Given a set of delegated prefixes, Nodes may assign available prefixes to links they are directly connected to, or for their private use. The algorithm ensures that the following assertions are satisfied after a finite convergence period:

1. At most one prefix from each delegated prefix is assigned to each link.

2. Assigned prefixes are non-overlapping (i.e., an assigned prefix never includes another assigned prefix).

3. Assigned prefixes do not change in the absence of topology or configuration changes.

In the rest of this document the two first conditions are referred to as the correctness conditions of the algorithm while the third condition is referred to as its convergence condition.

Each assignment has a priority specified by the Node making the assignment, allowing for custom assignment policies. When multiple Nodes assign different prefixes from the same delegated prefix to the same link, or when multiple Nodes assign overlapping prefixes (to the same link or to different links), the assignment with the greatest priority is kept and other assignments are removed.

The prefix assignment algorithm requires that participating Nodes share information through a flooding mechanism. If the flooding mechanism ensures that all messages are propagated to all Nodes within a given time window, the algorithm also ensures that all assigned prefixes used for networking operations (e.g., host configuration) remain unchanged, unless another Node assigns an overlapping prefix with a higher assignment priority, or the topology changes and renumbering cannot be avoided.

2. Terminology

In this document, the key words "MAY", "MUST, "MUST NOT", "OPTIONAL", and "SHOULD", are to be interpreted as described in [RFC2119].

This document makes use of the following terminology. The terms defined here are ordered in such a way as to avoid forward references, and therefore are not sorted alphabetically.

Node: An entity executing the algorithm specified in this document and able to communicate with other Nodes using the Flooding Mechanism.

Flooding Mechanism: A mechanism allowing participating Nodes to reliably share information with all other participating Nodes.

Link: An object the distributed algorithm will assign prefixes to. A Node may only assign prefixes to Links it is directly connected to. A Link is either Shared or Private.

Shared Link: A Link multiple Nodes may be connected to. Most of the time, a Shared Link is a multi-access link or point-to-point link, virtual or physical, requiring prefixes to be assigned to it.

Private Link: A Private Link is an abstract concept defined for the sake of this document. It allows Nodes to make assignments for their private use or delegation. For instance, every DHCPv6-PD [RFC3633] requesting router MAY be considered as a different Private Link.

Delegated Prefix: A prefix provided to the algorithm and used as a prefix pool for Assigned Prefixes.

Node ID: A value identifying a given participating Node. The set of identifiers MUST be strictly and totally ordered (e.g., using the alphanumeric order).

Flooding Delay: A value which MUST be provided by the Flooding Mechanism and SHOULD be a deterministic or likely upper bound on the information propagation delay among participating Nodes.

Advertised Prefix: A prefix advertised by another Node and delivered to the local Node by the Flooding Mechanism. It has an Advertised Prefix Priority and, when assigned to a directly connected Shared Link, is associated with that Shared Link.

Advertised Prefix Priority: A value that defines the priority of an Advertised Prefix received from the Flooding Mechanism or a published Assigned Prefix. Whenever multiple Advertised Prefixes are conflicting (i.e., overlapping or from the same Delegated Prefix and assigned to the same link), all Advertised Prefixes but the one with the greatest priority will eventually be removed. In case of a tie, the assignment advertised by the Node with the greatest Node ID is kept and others are removed. In order to ensure convergence, the range of priority values MUST have an upper bound.

Assigned Prefix: A prefix included in a Delegated Prefix and assigned to a Shared or Private Link. It represents a local decision to assign a given prefix from a given Delegated Prefix to a given Link. The algorithm ensures that there is never more than one Assigned Prefix per Delegated Prefix and Link pair. When destroyed, an Assigned Prefix is set as not applied, ceases to be advertised, and is removed from the set of Assigned Prefixes.

Applied (Assigned Prefix): When an Assigned Prefix is applied, it MAY be used (e.g., for host configuration, routing protocol configuration, prefix delegation). When not applied, it MUST NOT be used for any purpose outside of the prefix assignment algorithm. Each Assigned Prefix is associated with a timer (Apply Timer) used to apply the Assigned Prefix. An Assigned Prefix is unapplied when destroyed.

Published (Assigned Prefix): The Assigned Prefix is advertised through the Flooding Mechanism as assigned to its associated Link. A published Assigned Prefix MUST have an Advertised Prefix Priority. It will appear as an Advertised Prefix to other Nodes, once received through the Flooding Mechanism.

Prefix Adoption: When an Advertised Prefix which does not conflict with any other Advertised Prefix or published Assigned Prefix stops being advertised, any other Node connected to the same Link MAY, after some random delay, start advertising the same prefix. This procedure is called adoption and provides seamless assignment transfer from a Node to another, e.g., in case of Node failure.

Backoff Timer: Every Delegated Prefix and Link pair is associated with a timer counting down to zero. It is used to reduce the probability of colliding assignments made by multiple Nodes by delaying the creation of new Assigned Prefixes or the advertisement of adopted Assigned Prefixes by a random amount of time.

Renumbering: Event occurring when an Assigned Prefix which was applied is destroyed. Renumbering is undesirable as it usually implies reconfiguring routers or hosts.

2.1. Subroutine Specific Terminology

In addition to the terms defined in Section 2, the subroutine specified in Section 4 makes use of the following terms.

Current Assignment: For a given Delegated Prefix and Link, the Current Assignment is the Assigned Prefix (if any) included in the Delegated Prefix and assigned to the given Link by the Node executing the algorithm. At some point in time, the Current Assignment from different Nodes may differ, but the algorithm ensures that eventually, all Nodes directly connected to a Shared Link have the same Current Assignment for any given Delegated Prefix.

Precedence: An Advertised Prefix takes precedence over an Assigned Prefix if and only if one of the following conditions is met:

* The Assigned Prefix is not published.

* The Assigned Prefix is published and the Advertised Prefix Priority from the Advertised Prefix is strictly greater than the Advertised Prefix Priority from the Assigned Prefix.

* The Assigned Prefix is published, the priorities are identical, and the Node ID from the Node advertising the Advertised Prefix is strictly greater than the local Node ID.

Best Assignment: For a given Delegated Prefix and Link, the Best Assignment is the unique Advertised Prefix (if any) that:

* Includes or is included in the Delegated Prefix (i.e., the Advertised Prefix is a sub-prefix of the Delegated Prefix, or the Delegated Prefix is a sub-prefix of the Advertised Prefix).

* Is assigned on the given Link.

* Has the greatest Advertised Prefix Priority among Advertised Prefixes fulfilling the two preceding conditions (and, in case of a tie, the prefix advertised by the Node with the greatest Node ID among all prefixes with greatest priority).

* Takes precedence over the Current Assignment associated with the same Link and Delegated Prefix (if any).

Valid (Assigned Prefix): An Assigned Prefix is valid if and only if the following two conditions are met:

* No Advertised Prefix including or included in the Assigned Prefix takes precedence over the Assigned Prefix.

* No Advertised Prefix including or included in the same Delegated Prefix as the Assigned Prefix and assigned to the same Link takes precedence over the Assigned Prefix.

3. Applicability Statement

Each Node MUST have a set of non-overlapping Delegated Prefixes (i.e., which do not include each other). This set MAY change over time and be different from one Node to another at some point, but Nodes MUST eventually have the same set of disjoint Delegated Prefixes.

Given this set of disjoint Delegated Prefixes, Nodes may assign available prefixes from each Delegated Prefix to the Links they are directly connected to. The algorithm ensures that at most one prefix from a given Delegated Prefix is assigned to any given Link.

The algorithm can be applied to any address space and can be used to manage multiple address spaces simultaneously. For instance, an implementation can make use of IPv4-mapped IPv6 addresses [RFC4291]

in order to manage both IPv4 and IPv6 prefix assignment using a single prefix space.

The algorithm supports dynamically changing topologies:

- Nodes may join or leave the set of participating Nodes.
- Nodes may join or leave Links.
- Links may be joined or split.

All Nodes MUST run a common Flooding Mechanism in order to share published Assigned Prefixes. The set of participating Nodes is defined as the set of Nodes participating in the Flooding Mechanism.

The Flooding Mechanism MUST:

- Provide a way to flood Assigned Prefixes assigned to a directly connected Link along with their respective Advertised Prefix Priority and the Node ID of the Node which is advertising them.
- Specify whether an Advertised Prefix was assigned to a directly connected Shared Link, and if so, on which one.
- Provide a Flooding Delay value, which SHOULD represent a deterministic or likely upper bound on the information propagation delay among participating Nodes. Whenever the Flooding Mechanism is unable to adhere to the provided Flooding Delay, renumbering may happen. As such a delay often depends on the size of the network, it MAY change over time and MAY be different from one Node to another. Furthermore, the process of selecting this value is subject to a tradeoff between convergence speed and lower renumbering probability (e.g., the value 0 may be used when renumbering is harmless), and is therefore out of scope of this document.

The algorithm ensures that whenever the Flooding Delay is provided and respected, and in the absence of any topology change or Delegated Prefix removal, renumbering only happens when a Node deliberately overrides an existing assignment.

Each Node MUST have a Node ID. Node IDs MAY change over time and be the same on multiple Nodes at some point, but each Node MUST eventually have a Node ID which is unique among the set of participating Nodes.

4. Algorithm Specification

This section specifies the behavior of Nodes implementing the prefix assignment algorithm. The terms 'Current Assignment', 'Precedence', 'Best Assignment' and 'Valid' are used as defined in Section 2.1.

4.1. Prefix Assignment Algorithm Subroutine

This section specifies the prefix assignment algorithm subroutine. It is defined for a given Delegated Prefix and Link pair and takes a BackoffTriggered boolean as parameter (indicating whether the subroutine execution was triggered by the Backoff Timer or by another event).

For a given Delegated Prefix and Link pair, the subroutine MUST be run with the BackoffTriggered boolean set to false whenever:

- An Advertised Prefix including or included in the considered Delegated Prefix is added or removed.
- An Assigned Prefix included in the considered Delegated Prefix and associated with a different Link than the considered Link was destroyed, while there is no Current Assignment associated with the given pair. This case MAY be ignored if the creation of a new Assigned Prefix associated with the considered pair is not desired.
- The considered Delegated Prefix is added.
- The considered Link is added.
- The Node ID is modified.

Furthermore, for a given Delegated Prefix and Link pair, the subroutine MUST be run with the BackoffTriggered boolean set to true whenever:

- The Backoff Timer associated with the considered Delegated Prefix and Link pair fires while there is no Current Assignment associated with the given pair.

When such an event occurs, a Node MAY delay the execution of the subroutine instead of executing it immediately, e.g., while receiving an update from the Flooding Mechanism, or for security reasons (see Section 8). Even if other events occur in the meantime, the subroutine MUST be run only once. It is also assumed that, whenever one of these events is the Backoff Timer firing, the subroutine is executed with the BackoffTriggered boolean set to true.

In order to execute the subroutine for a given Delegated Prefix and Link pair, first look for the Best Assignment and Current Assignment associated with the Delegated Prefix and Link pair, then execute the corresponding case:

1. If there is no Best Assignment and no Current Assignment: Decide whether the creation of a new assignment for the given Delegated Prefix and Link pair is desired (As any result would be valid, the process of making this decision is out of the scope of this document) and do the following:

   * If it is not desired, stop the execution of the subroutine.

   * Else if the Backoff Timer is running, stop the execution of the subroutine.

   * Else if the BackoffTriggered boolean is set to false, set the Backoff Timer to some random delay between ADOPT_MAX_DELAY and BACKOFF_MAX_DELAY (see Section 7) and stop the execution of the subroutine.

   * Else, continue the execution of the subroutine.

   Select a prefix for the new assignment (see Section 5 for guidance regarding prefix selection). This prefix MUST be included in or be equal to the considered Delegated Prefix and MUST NOT include or be included in any Advertised Prefix. If a suitable prefix is found, use it to create a new Assigned Prefix:

   * Assigned to the considered Link.

   * Set as not applied.

   * The Apply Timer set to ’2 * Flooding Delay’.

   * Published with some selected Advertised Prefix Priority.

2. If there is a Best Assignment but no Current Assignment: Cancel the Backoff Timer and use the prefix from the Best Assignment to create a new Assigned Prefix:

   * Assigned to the considered Link.

   * Set as not applied.

   * The Apply Timer set to ’2 * Flooding Delay’.

   * Set as not published.

3. If there is a Current Assignment but no Best Assignment:

    * If the Current Assignment is not valid, destroy it, and execute the subroutine again with the BackoffTriggered boolean set to false.

    * If the Current Assignment is valid and published, stop the execution of the subroutine.

    * If the Current Assignment is valid and not published, the Node MUST either:

        + Adopt the prefix by canceling the Apply Timer and set the Backoff Timer to some random delay between 0 and ADOPT_MAX_DELAY (see Section 7). This procedure is used to avoid renumbering when the Node advertising the prefix left the Shared Link.

        + Destroy it and go to case 1.

4. If there is a Current Assignment and a Best Assignment:

    * Cancel the Backoff Timer.

    * If the two prefixes are identical, set the Current Assignment as not published. If the Current Assignment is not applied and the Apply Timer is not set, set the Apply Timer to '2 * Flooding Delay'.

    * If the two prefixes are not identical, destroy the Current Assignment and go to case 2.

When the prefix assignment algorithm subroutine requires an assignment to be created or adopted, any Advertised Prefix Priority value can be used. Other documents MAY provide restrictions over this value depending on the context the algorithm is operating in, or leave it as implementation-specific.

4.2. Overriding and Destroying Existing Assignments

In addition to the behaviors specified in Section 4.1, the following procedures MAY be used in order to provide additional behavior options (Section 6):

Overriding Existing Assignments: For any given Link and Delegated Prefix, a Node MAY create a new Assigned Prefix using a chosen prefix and Advertised Prefix Priority such that:

* The chosen prefix is included in or is equal to the considered Delegated Prefix.

* The Current Assignment, if any, as well as all existing Assigned Prefixes which include or are included inside the chosen prefix, are destroyed.

* It is not applied.

* The Apply Timer is set to '2 * Flooding Delay'.

* It is published.

* The Advertised Prefix Priority is greater than the Advertised Prefix Priority from all Advertised Prefixes which include or are included in the chosen prefix.

* The Advertised Prefix Priority is greater than the Advertised Prefix Priority from all Advertised Prefixes which include or are included in the considered Delegated Prefix and are assigned to the considered Link.

In order to ensure algorithm convergence:

* Such overriding assignments MUST NOT be created unless there was a change in the Node configuration, a Link was added, or an Advertised Prefix was added or removed.

* The chosen Advertised Prefix Priority for the new Assigned Prefix SHOULD be greater than all priorities from the destroyed Assigned Prefixes. If not, simple topologies with only two Nodes may not converge. Nodes which do not adhere to this rule MUST implement a mechanism which detects whether the distributed algorithm does not converge and, whenever this would happen, stop creating overriding Assigned Prefixes which do not adhere to this rule. The specifications for such safety procedures are out of the scope of this document.

Removing an Assigned Prefix: A Node MAY destroy any Assigned Prefix which is published. Such an event reflects the desire of a Node to not assign a prefix from a given Delegated Prefix to a given Link anymore. In order to ensure algorithm convergence, such a procedure MUST NOT be executed unless there was a change in the Node configuration. Furthermore, whenever an Assigned Prefix is destroyed in this way, the prefix assignment algorithm subroutine MUST be run for the Delegated Prefix and Link pair associated with the destroyed Assigned Prefix.

The two procedures specified in this section are OPTIONAL. They could be used for various purposes, e.g., for providing custom prefix assignment configuration or reacting to prefix space exhaustion (by overriding short Assigned Prefixes and assigning longer ones).

4.3. Other Events

When the Apply Timer fires, the associated Assigned Prefix MUST be applied.

When the Backoff Timer associated with a given Delegated Prefix and Link pair fires while there is a Current Assignment associated with the same pair, the Current Assignment MUST be published with some associated Advertised Prefix Priority and, if the prefix is not applied, the Apply Timer MUST be set to '2 * Flooding Delay'.

When a Delegated Prefix is removed from the set of Delegated Prefixes (e.g., when the Delegated Prefix expires), all Assigned Prefixes included in the removed Delegated Prefix MUST be destroyed.

When one Delegated Prefix is replaced by another one that includes or is included in the deleted Delegated Prefix, all Assigned Prefixes which were included in the deleted Delegated Prefix but are not included in the added Delegated Prefix MUST be destroyed. Others MAY be kept.

When a Link is removed, all Assigned Prefixes assigned to that Link MUST be destroyed.

5. Prefix Selection Considerations

When the prefix assignment algorithm subroutine specified in Section 4.1 requires a new prefix to be selected, the prefix MUST be selected either:

- Among prefixes included in the considered Delegated Prefix which were previously assigned and applied on the considered Link. For that purpose, Applied Prefixes may be stored in stable storage along with their associated Link.

- Randomly, picked in a set of at least RANDOM_SET_SIZE (see Section 7) prefixes included in the considered Delegated Prefix and not including or included in any Assigned or Advertised Prefix. If less than RANDOM_SET_SIZE candidates are found, the prefix MUST be picked among all candidates.

- Based on some custom selection process specified in the configuration.

A simple implementation MAY randomly pick the prefix among all available prefixes, but this strategy is inefficient in terms of address space use as a few long prefixes may exhaust the pool of available short prefixes.

The rest of this section describes a more efficient approach which MAY be applied any time a Node needs to pick a prefix for a new assignment. The two following definitions are used:

Available prefix: The prefix of the form Prefix/PrefixLength is available if and only if it satisfies the three following conditions:

* It is included in the considered Delegated Prefix.

* It does not include and is not included in any Assigned or Advertised Prefix.

* It is equal to the considered Delegated Prefix or Prefix/(PrefixLength-1) includes an Assigned or Advertised Prefix.

Candidate prefix: A prefix of desired length which is included in or is equal to an available prefix.

The procedure described in this section takes the three following criteria into account:

Prefix Stability: In some cases, it is desirable that the selected prefix should remain the same across executions and reboots. For this purpose, prefixes previously applied on the Link or pseudo-random prefixes generated based on Node- and Link-specific values may be considered.

Randomness: When no stored or pseudo-random prefix is chosen, a prefix may be randomly picked among RANDOM_SET_SIZE candidates of desired length. If less than RANDOM_SET_SIZE candidates can be found, the prefix is picked among all candidates.

Addressing-space usage efficiency: In the process of assigning prefixes, a small set of badly chosen long prefixes may prevent any shorter prefix from being assigned. For this reason, the set of RANDOM_SET_SIZE candidates is created from available prefixes with longest prefix lengths and, in case of a tie, preferring numerically small prefix values.

When executing the procedure, do as follows:

1. For each prefix stored in stable storage, check if the prefix is included in or equal to an available prefix. If so, pick that prefix and stop.

2. For each prefix length, count the number of available prefixes of the given length.

3. If the desired prefix length was not specified, select one. The available prefixes count computed previously may be used to help pick a prefix length such that:

   * There is at least one candidate prefix.

   * The prefix length is chosen large enough to not exhaust the address space.

   Let N be the chosen prefix length.

4. Iterate over available prefixes starting with prefixes of length N down to length 0 and create a set of RANDOM_SET_SIZE candidate prefixes of length exactly N included in or equal to available prefixes. The end goal here is to create a set of RANDOM_SET_SIZE candidate prefixes of length N included in a set of available prefixes of maximized prefix length. In case of a tie, smaller prefix values (as defined by the bit-wise lexicographical order) are preferred.

5. Generate a set of prefixes of desired length, which are pseudo-randomly chosen based on Node- and Link-specific values. For each pseudo-random prefix, check if the prefix is equal to a candidate prefix. If so, pick that prefix and stop.

6. Choose a random prefix from the set of selected candidates.

The complexity of this procedure is equivalent to the complexity of iterating over available prefixes. Such operation may be accomplished in linear time, e.g., by storing Advertised and Assigned Prefixes in a binary trie.

6. Implementation Capabilities and Node Behavior

Implementations of the prefix assignment algorithm may vary from very basic to highly customizable, enabling different types of fully interoperable behaviors. The three following behaviors are given as examples:

Listener: The Node only acts upon assignments made by other Nodes, i.e, it never creates new assignments nor adopts existing ones.

Such behavior does not require the implementation of the considerations specified in Section 5 or Section 4.2. The Node never checks the validity of existing assignments, which makes this behavior particularly suited to lightweight devices which can rely on more capable neighbors to make assignments on directly connected Shared Links.

Basic: The Node is capable of assigning new prefixes or adopting prefixes which do not conflict with any other existing assignment. Such behavior does not require the implementation of the considerations specified in Section 4.2. It is suited to situations where there is no preference over which prefix should be assigned to which Link, and there is no priority between different Links.

Advanced: The Node is capable of assigning new prefixes, adopting existing ones, making overriding assignments and destroying existing ones. Such behavior requires the implementation of the considerations specified in Section 5 and Section 4.2. It is suited when the administrator desires some particular prefix to be assigned on a given Link, or some Link to be assigned prefixes with a greater priority when there are not enough prefixes available for all Links.

Note that if all Nodes directly connected to some Link are listener Nodes or none of these Nodes are willing to make an assignment from a given Delegated Prefix to the given Link, no prefix from the given Delegated Prefix will ever be assigned to the Link (and such existing prefixes will be removed). This situation may be detected by watching whether no prefix from a given Delegated Prefix has been assigned to the Link for longer than BACKOFF_MAX_DELAY plus the Flooding Delay.

7. Algorithm Parameters

This document does not provide values for ADOPT_MAX_DELAY, BACKOFF_MAX_DELAY and RANDOM_SET_SIZE. The algorithm ensures convergence and correctness for any chosen values, even when these are different from Node to Node. They MAY be adjusted depending on the context, providing a tradeoff between convergence time, efficient addressing, reduced control traffic (generated by the Flooding Mechanism), and low collision probability.

ADOPT_MAX_DELAY (respectively BACKOFF_MAX_DELAY) represents the maximum backoff time a Node may wait before adopting an assignment (respectively making a new assignment). BACKOFF_MAX_DELAY MUST be greater than or equal to ADOPT_MAX_DELAY. The greater ADOPT_MAX_DELAY and (BACKOFF_MAX_DELAY - ADOPT_MAX_DELAY), the lower

the collision probability and the lesser the amount of control traffic, but the greater the convergence time.

RANDOM_SET_SIZE represents the desired size of the set a random prefix will be picked from. The greater RANDOM_SET_SIZE, the better the convergence time and the lower the collision probability, but the worse the addressing-space usage efficiency.

8. Security Considerations

The prefix assignment algorithm functions on top of two distinct mechanisms, the Flooding Mechanism and the Node ID assignment mechanism.

An attacker able to publish Advertised Prefixes through the Flooding Mechanism may perform the following attacks:

* Publish a single overriding assignment for a whole Delegated Prefix or for the whole address space, thus preventing any Node from assigning prefixes to Links.

* Quickly publish and remove Advertised Prefixes, generating traffic at the Flooding Mechanism layer and causing multiple executions of the prefix assignment algorithm in all participating Nodes.

* Publish and remove Advertised Prefixes in order to prevent the convergence of the algorithm.

An attacker able to prevent other Nodes from accessing a portion or the whole set of Advertised Prefixes may compromise the correctness of the algorithm.

An attacker able to cause repetitive Node ID changes may cause traffic to be generated in the Flooding Mechanism and multiple executions of the prefix assignment algorithm in all participating Nodes.

An attacker able to publish Advertised Prefixes using a Node ID used by another Node may prevent the correctness and convergence of the algorithm or cause the result to violate the correctness conditions.

Whenever the security of the Flooding Mechanism and Node ID assignment mechanism cannot be ensured, the convergence of the algorithm may be prevented. In environments where such attacks may be performed, the execution of the prefix assignment algorithm subroutine SHOULD be rate limited, as specified in Section 4.1.

9. IANA Considerations

This document has no actions for IANA.

10. Acknowledgments

The authors would like to thank those who participated in the previous document's version development as well as the present one. In particular, the authors would like to thank Tim Chown, Fred Baker, Mark Townsley, Lorenzo Colitti, Ole Troan, Ray Bellis, Markus Stenberg, Wassim Haddad, Joel Halpern, Samita Chakrabarti, Michael Richardson, Anders Brandt, Erik Nordmark, Laurent Toutain, Ralph Droms, Acee Lindem and Steven Barth for interesting discussions and document review.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

11.2. Informative References

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

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

Appendix A. Static Configuration Example

This section describes an example of how custom configuration of the prefix assignment algorithm may be implemented.

The Node configuration is specified as a finite set of rules. A rule is defined as:

- A prefix to be used.
- A Link on which the prefix may be assigned.
- An Assigned Prefix Priority (smallest possible Assigned Prefix Priority if the rule may not override other Assigned Prefixes).
- A rule priority (0 if the rule may not override existing Advertised Prefixes).

In order to ensure the convergence of the algorithm, the Assigned Prefix Priority MUST be an increasing function (not necessarily strictly) of the configuration rule priority (i.e., the greater is the configuration rule priority, the greater the Assigned Prefix Priority must be).

Each Assigned Prefix is associated with a rule priority. Assigned Prefixes which are created as specified in Section 4.1 are given a rule priority of 0.

Whenever the configuration is changed or the prefix assignment algorithm subroutine is run: For each Link/Delegated Prefix pair, look for the configuration rule with the greatest configuration rule priority such that:

- The prefix specified in the configuration rule is included in the considered Delegated Prefix.
- The Link specified in the configuration rule is the considered Link.
- All the Assigned Prefixes which would need to be destroyed in case a new Assigned Prefix is created from that configuration rule (as specified in Section 4.2) have an associated rule priority which is strictly lower than the one of the considered configuration rule.
- The assignment would be valid when published with an Advertised Prefix Priority equal to the one specified in the configuration rule.

If a rule is found, a new Assigned Prefix is created based on that rule as specified in Section 4.2. The new Assigned Prefix is associated with the Advertised Prefix Priority and the rule priority specified in the considered configuration rule.

Note that the use of rule priorities ensures the convergence of the algorithm.

Authors' Addresses

Pierre Pfister
Cisco Systems
Paris
France

Email: pierre.pfister@darou.fr

Benjamin Paterson
Cisco Systems
Paris
France

Email: paterson.b@gmail.com

Jari Arkko
Ericsson
Jorvas 02420
Finland

Email: jari.arkko@piuha.net

Home Networking
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

K. Jin
Ecole Polytechnique / Cisco Systems
P. Pfister
Cisco Systems
J. Yi
LIX, Ecole Polytechnique

July 6, 2015

Experience and Evaluation of the Distributed Node Consensus Protocol
draft-jin-homenet-dncp-experience-00

Abstract

The Distributed Node Consensus Protocol (DNCP) is a generic state synchronization protocol that offers data synchronization and dynamic network topology discovery within a network. This document reports experience with the DNCP, which includes its implementation status and performance evaluation in a simulated environment.

Table of Contents

1. Introduction

The Distributed Node Consensus Protocol (DNCP) [I-D.ietf-homenet-dncp] is a protocol providing node data synchronization and dynamic network topology discovery within a network. At the time of writing this document, DNCP is in standardization process by the IETF Homenet working group.

In DNCP, the information of a node and its view of the network is encoded in a set of TLV (Type-Length-Value) tuples. By publishing and exchanging the TLVs with its neighbors, each node in the network eventually receives the set of TLV published by every other node within the network (in which case, the network is converged). The Trickle algorithm [RFC6206] is used, instead of periodic signaling, to reduce the overhead of synchronization. DNCP also provides an option of "keep alive" message to detect when a neighbor is not reachable anymore.

As a generic protocol, DNCP can not only be applied in home networks, but also in other networks that require node data synchronization (such as configuration profiles, network topology, services, etc.). Therefore, DNCP leaves some parameters to be specified by DNCP profiles, which are actual implementable instances of DNCP. Nodes implementing the same DNCP profile, and operating within the same DNCP network, are able share TLV tuples, discover the network topology, and auto-detect arrival and departure of other nodes.

This document presents experience and performances evaluation using the reference DNCP implementation in a simulated environment using the Network Simulator 3 (NS3), a discrete event simulator widely used in network research and recognized by the scientific community.

Note that for the purpose of this first study, DNCP was evaluated in various, quite unrealistic and probably extreme, network topologies, with the intent of finding the limits of the protocol.

2. Implementation Status

Until July 2015, there are 2 publicly known implementations of DNCP, one of which was recently modified.

2.1. hnetd Implementation (libdncp)

'hnetd' is an open source implementation of the Homenet network stack. As an implementation of the Home Network Control Protocol - HNCP [I-D.ietf-homenet-hncp], it includes the DNCP, the prefix assignment algorithm [I-D.ietf-homenet-prefix-assignment], as well as

other elements specific to Homenet. hnetd is available as a package to be installed on OpenWrt, which is the platform it is most suited for (hnetd also runs in a lesser extend on standard Linux). The code is available on github [1] and is published under the GPLv2 license (libdncp is available in the 'libdncp' git branch).

hnetd is based on a lightweight toolkit library containing useful structures (e.g., lists, trees), functions (e.g., md5), as well as a small event loop, that is widely used in OpenWrt. Thanks to the quality of the code, libdncp, as separatly buildable library implementing DNCP functions, was easily extracted from hnetd for the purpose of this work.

hnetd, DNCP included, consists of 15651 lines of code (18220 when including test files). The binary weights 576KB when compiled for debian X86_64 with no optimization and 727KB when compiled for OpenWrt MIPS. libdncp2 roughly consists of 2300 lines of code (2590 when including security option). It weights 193KB when compiled with no optimization for debian x86_64 and 192KB when compiled for OpenWrt MIPS.

2.2. libdncp2 Implementation

As a result of different interests expressed about using DNCP outside of Homenet (including this study), DNCP code within hnetd was partly rewritten and reorganized in a more independent implementation of DNCP, with less coupling with HNCP [I-D.ietf-homenet-hncp]. libdncp2 moves the DNCP profiles specificities from compilation-time to run-time. It is published under the same license as hnetd and is now part of the mainstream branch [1].

libdncp2 provides some improvements with respect to its predecessor libdncp. For the purpose of this document, libdncp was used, but we plan to use libdncp2 instead in the next iterations of this document.

2.3. shncpd Implementation

shncpd is an open source implementation of DNCP developed by Juliusz Chroboczek. It uses HNCP [I-D.ietf-homenet-hncp] profile. The source code is available on github [2]. At the time of publishing this document, shncpd implements:

- The HNCP profile of DNCP.
- Parsing of a significant subset of HNCP.
- Address allocation (not prefix allocation).

3. Experimental Setup

This section describes the environment, the parameters and topologies which were used for measuring DNCP performances.

3.1. Simulation Environment

For the purpose of this work, the DNCP part (libdncp) of'hnetd' introduced in Section 2.1 was modified in order to provide a statically linkable library containing DNCP implementation. To evaluate the performance of DNCP in large and complex networks, NS3 is employed as the simulation platform.

As shown in Figure 1, all the application-level actions (processing packets, publishing TLVs...) are performed inside the libdncp implementation. The packets are sent to or received from the lower layers in ns3 through the redefined socket API. UDP is used for layer 4 and IPv6 for layer 3. NS3 implements different types of links as layer 1 and layer 2. For these measures, the so called 'CSMA' link type is used. The NS3 CSMA link is designed in the spirit of Ethernet but implements fail-proof carrier sense used for collision avoidance. For this first study, where measuring DNCP link usage was desired, link of virtually infinite throughput are used.

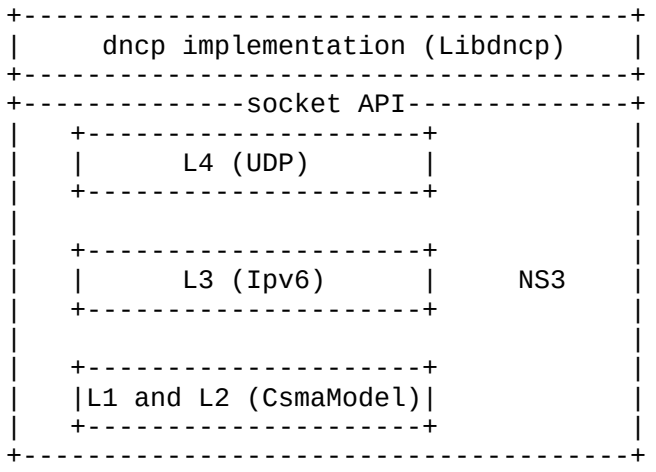


Figure 1: libdncp over NS3 network layering model

Listed below are several attributes of the CSMA model:

MTU: The link layer maximum transmission unit, set to 1500.

Encapsulation Mode: Type of link layer encapsulation. "Dix" mode was used, which models an Ethernet II layer.

TxQueue: Type of the transmit queue used by the device. NS3 provides "Codel queue", "Drop tail" and "RED" (random early detection) queues. "Drop tail" queue was used with a size of 100 packets.

Inter-frame Gap: The pause between two frames, set to 0.

Listed below are several attributes of the CSMA channel:

Data Rate: Physical layer bit rate, enforcing the time it takes for a frame to be transmitted. It is set to 1Gbps, which is significantly greater than what DNCP is expected to use.

Delay: Signal propagation time within the medium. It was set to 1 micro second.

Assuming a constant frame size, the theoretical throughput of the medium is given by the formula FrameSize/(FrameSize/DataRate + Delay). For a frame size of 1500 bytes, the throughput is 923Mbps. For a FrameSize of 100 bytes, the link throughput is 444Mbps.

Running DNCP in NS3 requires libdncp to be integrated and built with NS3. DNCP runs on an event loop managed by libubox, which therefore specifies how to set timers and listen to file descriptors. Integration was quite straitforward. Event-loop and system calls were identified and replaced with their NS3 equivalents. Besides, an application in ns3 called "DncpApplication" is created, this application can be installed on the node and can be started and stopped at given time. Once the application is launched, dncp begins to run on that node by calling functions in the libdncp static library from inside the ns3 application. It is expected that integration with libdncp2 will be even simpler, as this new library put the DNCP profile definition at runtime instead of compilation time.

Running experiments in simulated environments offers multiple advantages such as the ability to run long-lived scenarios in short period of time, simulate networks of hundread of nodes without requiring lots of resources, or isolate tested components from external interferences. On the other hand, NS3 executes program steps virtually instantaneously, it is therefore hard to take into account hardware speed when measuring time-related performances metrics. For these first experiments, virtually perfect links with

no delay were used, and a processing delay of 0.5ms for each received packet was introduced in order to simulate the packet processing time.

3.2. Performance Metrics

The goal of this study is to measure the performances of DNCP in some extreme scenarios, see whether, and how it converges. Therefore, the following three performance metrics were observed:

Convergence time: The time it took for the network to converge.

Overall traffic sent: The amount of data that was sent on link before convergence.

Average traffic sent per node: The overall traffic sent divided by the number of nodes.

3.3. Chosen Toplogies

This section describes different topologies used in this study. The main criteria for selecting a topology was that it should be:

- Easily described and generated as a function of the number of nodes (called N).
- Deterministic.
- Representing different situations testing different scalability properties.

The rest of this section describes the five different topologies:

- Single link
- String
- Full mesh
- Tree topology
- Double tree topology

3.3.1. Single Link Topology

The single link topology puts all the nodes on the same link. Each node has a single DNCP End-Point with N-1 neighbors. Such topology is well suited for evaluating DNCP scalability capabilities in terms

of the number of neighbors on a given link.

Figure 2: The single link topology for N = 4

3.3.2. String Topology

The string topology chains all nodes one after the other. Each node has two DNCP End-Points with one neighbor on each (except for the two extremities). Such topology is well suited for evaluating the convergence time depending on the diameter of the network as well as the scalability of DNCP in terms of the overall number of nodes.

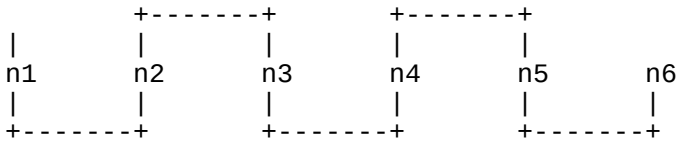


Figure 3: The string topology for N = 6

3.3.3. Full Mesh Topology

The mesh topology connects all nodes with distinct links. Each node has N-1 DNCP End-Points with one neighbor on each. Such topology is well suited for evaluating DNCP scalability capabilities in terms of the amount of nodes and end-points.

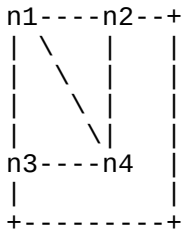


Figure 4: The full mesh topology for N = 4

3.3.4. Tree Topology

The tree topology forms a typical binary tree. Node 'i' is connected with nodes '2*i' and '2*i + 1', unless those numbers are greater than N. In such topology, all nodes except the root one have three DNCP End-Points with one neighbor on each. This topology offers a more realistic equilibrium between the diameter and the amount of nodes.

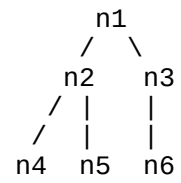


Figure 5: The tree topology for N = 6

3.3.5. Double Tree Topology

The double tree topology is identical to the binary tree, but each node is paired with a redundancy node. In such topology, all nodes except the two root node have 6 DNCP End-Point with one neighbor on each. This topology also offers a more realistic trade-off between the network diameter and the number of nodes, but also adds redundancy and loops.

For example, for N = 9:

- n1 has point-to-point links with n3, n4, n5 and n6.
- n2 has point-to-point links with n3, n4, n5 and n6.
- n3 has additional point-to-point links with n7, n8 and n9.
- n4 has additional point-to-point links with n7, n8 and n9.

4. DNCP Performance Evaluation

This section provides different performance metrics for different network topologies of different sizes. Each value is the average over 10 simulations. Unless stated otherwise, the 10 measures always were pretty close (Small standard deviation).

Each simulation reflects the same scenario. All DNCP instances are simultaneously initialized. The simulation is then run until the network converges, and performance metrics introduced in Section 3.2

are computed.

Important: Note that, as DNCP uses Trickle, it is expected to be very verbose in case of dramatic changes, but as a trade-off should provide good convergence times and low verbosity in the absence of changes. An instantaneous and complete bootstrap of the network, which is the present scenario, is a particularly extreme state change. This scenario was selected as a worst case to see whether DNCP converges well, or at-all. On that side, DNCP behaved well, but as expected, the overall amount of generated traffic sometimes appeared to be significant.

4.1. Results for the Single Link Topology

Number of nodes	10 nodes	20 nodes	30 nodes	40 nodes
Convergence time	1.84s	3.09s	*4.43s	5.14s
Traffic sent	85.3KB	604.7KB	2.3MB	5.4MB
Traffic sent per node	8.5KB	30.2KB	79.6KB	140.7KB

Table 1: Single Link Topology (1)

Number of nodes	50 nodes	60 nodes	70 nodes	80 nodes
Convergence time	6.53s	*8.61s	11.57s	14.05s
Traffic sent	11.9MB	23.7MB	51.7MB	88.1MB
Traffic sent per node	245.KB	404.8KB	757.2KB	1.1MB

*: the average value was calculated over the results of 9 experiments because one of the value was significantly greater.

Table 2: Single Link Topology (2)

Note that two accidents were observed during the simulations. One happened in one experiment among the 10 that we ran for the 30-nodes network. The network first converged at 4.016s, which is very close to the average convergence time, but at 25.949s this converging state was broken and the network finally reconverged at 26.12s. Similarly, for the 60-node network, it first converged at 7.081s then got disturbed at 25.822s and converged again at 26.303.

Although the convergence time seems to grow linearly with the number of nodes, the overall traffic sent before convergence increases dramatically. This is caused by the fact that not-only each node will synchronize with any other node on the link, but the data will grow as nodes discover their neighbors. The problem is explained in [I-D.ietf-homenet-dncp] Section 6.2 (Support For Dense Broadcast Links) where an optional solution is also proposed, but that option *was not* implemented in libdncp, which explains the bad performances.

4.2. Results for the String Topology

Number of nodes	10 nodes	20 nodes	30 nodes	40 nodes
Convergence time	1.84s	3.65s	5.24s	7.09s
Traffic sent	51.5KB	243.4KB	605KB	1.2MB
Traffic sent per node	5.1KB	12.2KB	20.1KB	30.9KB

Table 3: String topology (1)

Number of nodes	50 nodes	60 nodes	70 nodes	80 nodes
Convergence time	8.79s	11.11s	12.87s	15.03s
Traffic sent	2MB	3MB	4.1MB	5.6MB
Traffic sent per node	40.5KB	50.4KB	59.2KB	70.1KB

Table 4: String topology (2)

These results show that in a linear network, convergence time is linear in the number of chained nodes, but the overall transmitted traffic is not. This can be easily explained as the convergence time reflects the time for both extremities to discover each other (updates have to traverse the whole string), but in the meantime, other updates are sent. The slope of the convergence time line is 0.18 second per node, which is pretty quick, but also comes at the cost of transmitting multiple updates before all nodes are discovered.

4.3. Results for the full mesh topology

Number of nodes	10 nodes	20 nodes	30 nodes	40 nodes
Convergence time	1.71s	3.2s	4.83s	*6.19s
Traffic sent	202.7KB	1.6MB	6.6MB	18.1MB
Traffic sent per node	20.3KB	83.5KB	222.1KB	453.8KB

*: the average value was calculated over the results of 9 experiments because one of the value was significantly greater.

Table 5: Full mesh topology (1)

Number of nodes	50 nodes	60 nodes	70 nodes	80 nodes
Convergence time	10.64s	13.02s	15.33s	17.93s
Traffic sent	49.1MB	95.8MB	167.4MB	271.9MB
Traffic sent per node	983.1KB	1.6MB	2.4MB	3.4MB

Table 6: Full mesh topology (2)

An incident similar to the one that occurred with a single link topology was observed. Although the convergence time is low, the amount of transmitted traffic is very high compared to other topologies. It can be explained by the high number of distinct links (equal to N(N-1)/2): Trickle has to converge on each individual link. Unlike the single link topology, this situation is harder to detect and resolve.

4.4. Results for the tree topology

Number of nodes	10 nodes	20 nodes	30 nodes	40 nodes
Convergence time	1.16s	1.57s	1.86s	2s
Traffic sent	40.7KB	166.7KB	374KB	644.5KB
Traffic sent per node	4.1KB	8.3KB	12.4KB	16.1KB

Table 7: Tree topology (1)

Number of nodes	50 nodes	60 nodes	70 nodes	80 nodes
Convergence time	2.33s	2.42s	2.56s	2.6s
Traffic sent	1MB	1.3MB	1.9MB	2.4MB
Traffic sent per node	20.2KB	22.8KB	26.7KB	29.9KB

Table 8: Tree topology (2)

As expected in a tree topology, the convergence time is sub-linear. We also oberve that the overall amount of traffic (and per node) is relatively low compared to other topologies. This is quite satisfying as the tree and double-tree topoligies are the more realistic ones.

4.5. Results for the double-tree topology

Number of nodes	10 nodes	20 nodes	30 nodes	40 nodes
Convergence time	1.04s	1.44s	1.5s	1.7s
Traffic sent	66.9KB	265KB	605.1KB	1MB
Traffic sent per node	6.7KB	13.2KB	20.2KB	25.3KB

Table 9: Double-tree topology (1)

Number of nodes	50 nodes	60 nodes	70 nodes	80 nodes
Convergence time	1.96s	1.98s	2.06s	2.09s
Traffic sent	1.5MB	2MB	2.8MB	3.5MB
Traffic sent per node	30.8KB	33.2KB	39.7KB	44.7KB

Table 10: Double-tree topology (2)

Results are quite similar to the simple tree topology. The convergence delay is very satisfying while the overall amount of traffic is pretty low compared to other topologies.

5. Conclusion

DNCP does converge in small networks as well as larger ones. It converges fast at the cost of a quite high overall transmitted amount of data. It behaves particularly well in tree topologies as well as double tree topologies, which are the most realistic tested topologies.

The first observed concern appears in the single link topology, where the amount of traffic grows dramatically with the number of nodes. The problem is known and DNCP actually proposes a solution to that problem. But this solution is optional and was not part of the tested implementation. Further attention may be necessary on these particular mechanisms in order to make sure that DNCP behaves well in such situations (if such topology is in scope of DNCP at all).

The overall amount of traffic was also significant in full mesh topologies as well as string topologies. A possible approach to improve these results might be to rate-limit the amount of updates

that may be made in short periods of time. Such approach would provide fast convergence after small changes and would reduce the overall amount of traffic in reaction to dramatic changes.

Once again, it is important to note that DNCP is expected to be verbose in case of regular or dramatic changes. DNCP users should make sure that the network eventually stabilizes, such that DNCP can take full advantage of the Trickle algorithm.

Next iterations of this document might include further results such as:

- Measures with libdncp2 instead of libdncp(v1).
- Measures with different profiles (rather than the HNCP profile alone).
- Other metrics (e.g., convergence ratio).
- Other scenarios (e.g., no updates at all, sporadic updates).

6. Security Considerations

This document does not specify a protocol or a procedure. DNCP's security architecture is described in Section 8 of [I-D.ietf-homenet-dncp].

7. IANA Considerations

This document contains no action for IANA.

8. Informative References

[I-D.ietf-homenet-dncp]
Stenberg, M. and S. Barth, "Distributed Node Consensus Protocol", draft-ietf-homenet-dncp-07 (work in progress), July 2015.

[I-D.ietf-homenet-hncp]
Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", draft-ietf-homenet-hncp-07 (work in progress), July 2015.

[I-D.ietf-homenet-prefix-assignment]
Pfister, P., Paterson, B., and J. Arkko, "Distributed

Prefix Assignment Algorithm",
draft-ietf-homenet-prefix-assignment-07 (work in
progress), June 2015.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko,
"The Trickle Algorithm", RFC 6206, March 2011.

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

[2] <https://github.com/jech/shncpd>

Appendix A. Acknowledgments

The authors would like to thank Markus Stenberg for his help with hnetd and the remarkable quality of his code, as well as Juliusz Chroboczek for the new (yet untested) DNCP implementation.

Authors' Addresses

Kaiwen Jin
Ecole Polytechnique / Cisco Systems
Palaiseau,
France

Email: kaiwen.jin@master.polytechnique.org

Pierre Pfister
Cisco Systems
Paris,
France

Email: pierre.pfister@darou.fr

Jiazi Yi
LIX, Ecole Polytechnique
Route de Saclay
Palaiseau 91128,
France

Phone: +33 1 77 57 80 85
Email: jiazi@jiaziyi.com
URI: http://www.jiaziyi.com/