Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 09, 2014

M. Stenberg

S. Barth

February 05, 2014

Home Networking Control Protocol
draft-stenberg-homenet-hncp-00

Abstract

This document describes the HomeNet Control Protocol (HNCP), a minimalist state synchronization protocol for Homenet routers.

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 August 09, 2014.

Table of Contents

1. Introduction

HNCP is designed to synchronize state across a Homenet (or other small site) in order to facilitate automated configuration within the site, integration with trusted bootstrapping [I-D.behringer-homenet-trust-bootstrap] and default perimeter detection [I-D.kline-homenet-default-perimeter], automatic IP prefix distribution [I-D.pfister-homenet-prefix-assignment], and service discovery across multiple links within the homenet as defined in [I-D.stenberg-homenet-dnssd-hybrid-proxy-network-zeroconf].

HNCP is designed to provide enough information for a routing protocol to operate without homenet-specific extensions. In homenet environments where multiple IPv6 prefixes are present, routing based on source and destination address is necessary [I-D.troan-homenet-sadr]. Routing protocol requirements for source and destination routing are described in section 3 of [I-D.baker-rtgwg-src-dst-routing-use-cases].

A GPLv2-licensed implementation of the HNCP protocol is currently under development at https://github.com/sbyx/hnetd/ [1]. Comments and/or pull requests are welcome.

An earlier implementation using many of the same principles, algorithms and data structures built within OSPFv3 is available at http://www.homewrt.org/doku.php?id=downloads [2].

2. Requirements language

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

3. Data model

The data model of the HNCP protocol is simple: Every participating node has (and also knows for every other participating node):

A unique node identifier. It may be a public key, unique hardware ID, or some other unique blob of binary data which HNCP can run a hash upon to obtain a node identifer that is very likely unique among the set of routers in the Homenet.

A set of Type-Length-Value (TLV) data it wants to share with other routers. The set of TLVs have a well-defined order based on ascending binary content that is used to quickly identify changes in the set as they occur.

Latest update sequence number. A four octet number that is incremented anytime TLV data changes are detected.

Relative time, in milliseconds, since last publishing of the current TLV data set.

If HNCP security is enabled, each node will have a public/private key pair defined. The private key is used to create signatures for messages and node state updates and never sent across the network by HNCP. The public key is used to verify signatures of messages and node state updates.

4. Operation

HNCP is designed to run on UDP port IANA-UDP-PORT, using both link-local scoped IPv6 unicast and link-local scoped IPv6 multicast messages to address IANA-MULTICAST-ADDRESS for transport. The protocol consists of Trickle [RFC6206] driven multicast status messages to indicate changes in shared TLV data, and unicast state synchronization message exchanges when the Trickle state is found to be inconsistent.

4.1. Trickle-Driven Status Updates

Each node MUST send link-local multicast NetState Messages (Section 4.2.1) each time the Trickle algorithm [RFC6206] indicates they should on each link the protocol is active on. When the locally stored network state hash changes (either by a local node event that affects the TLV data, or upon receipt of more recent data from another node), all Trickle instances MUST be reset. Trickle state MUST be maintained separately for each link.

Trickle algorithm has 3 parameters; Imin, Imax and k. Imin and Imax represent minimum and maximum values for I, which is the time interval during which at least k Trickle updates must be seen on a link to prevent local state transmission. Bounds for recommended Trickle values are described below.

k=1 SHOULD be used, as given the timer reset on data updates, retransmissions should handle packet loss.

Imax MUST be at least one minute.

Imin MUST be at least 200 milliseconds (earliest transmissions may occur at Imin/2 = 100 milliseconds given minimum values as per the Trickle algorithm).

4.2. Protocol Messages

Protocol messages are encoded as purely as a sequence of TLV objects (Section 5). This section describes which set of TLVs MUST or MAY be present in a given message.

In order to facilitate fast comparing of local state with that in a received message update, all TLVs in every encoding scope (either root level, within the message itself, or within a container TLV) MUST be placed in ascending order based on the binary comparison of both TLV header and value. By design, the TLVs which MUST be present have the lowest available type values, ensuring they will naturally occur at the start of the Protocol Message, resembling a fixed format preamble.

4.2.1. Network State Update (NetState)

This Message SHOULD be sent as a multicast message.

The following TLVs MUST be present at the start of the message:

Node Link TLV (Section 5.2.1).

Network State TLV (Section 5.2.2).

The NetState Message MAY contain Node State TLV(s) (Section 5.2.3). If so, either all Node State TLVs are included (referred to as a "long" NetState Message), or none are included (refered to as a "short" NetState Message). The NetState Message MUST NOT contain only a portion of Node State TLVs as this could cause problems with the Protocol Message Processing (Section 4.3) algorithm. Finally, if the long version of the NetState message would exceed the minimum IPv6 MTU when sent, the short version of the NetState message MUST be used instead.

If HNCP security is enabled, authentication TLVs (Section 5.4) MUST be present.

4.2.2. Network State Request, (NetState-Req)

This Message MUST be sent as a unicast message.

The following TLVs MUST be present at the start of the message:

Node Link TLV (Section 5.2.1).

Request Network State TLV (Section 5.1.1).

If HNCP security is enabled, authentication TLVs (Section 5.4) MUST be present.

4.2.3. Node Data Request (Node-Req)

This Message MUST be sent as a unicast message.

MUST be present:

Node Link TLV (Section 5.2.1).

one or more Request Node Data TLVs (Section 5.1.2).

If HNCP security is enabled, authentication TLVs (Section 5.4) MUST be present.

4.2.4. Network and Node State Reply (NetNode-Reply)

This Message MUST be sent as a unicast message.

MUST be present:

Node Link TLV (Section 5.2.1).

Network State TLV (Section 5.2.2) and Node State TLV (Section 5.2.3) for every known node by the sender, or

one or more combinations of Node State and Node Data TLVs (Section 5.2.4).

If HNCP security is enabled, authentication TLVs (Section 5.4) MUST be present.

4.3. HNCP Protocol Message Processing

The majority of status updates among known nodes are handled via the Trickle-Driven updates (Section 4.1). This section describes processing of messages as received, along with associated actions or responses.

HNCP is designed to operate between directly connected neighbors on a shared link using link-local IPv6 addresses. If the source address of a received HNCP packet is not an IPv6 link-local unicast address, the packet SHOULD be dropped. Similarly, if the destination address

is not IPv6 link-local unicast or IPv6 link-local multicast address, packet SHOULD be dropped.

Upon receipt of:

NetState Message (Section 4.2.1): If the network state hash within the message matches the hash of the locally stored network state, consider Trickle state as consistent with no further processing required. If the hashes do not match, consider Trickle state as inconsistent. In this case, if the message is "short" (contains zero Node State TLVs), reply with a NetState-Req Message (Section 4.2.2). If the message was in long format (contained all Node State TLVs), reply with NodeState-Req (Section 4.2.3) for any nodes for which local information is outdated (local update number is lower than that within the message) or missing. Note that if local information is more recent than that of the neighbor, there is no need to send a message.

NetState-Req (Section 4.2.2): Provide requested data in a NetNode-Reply Message containing Network State TLV and all Node State TLVs.

NodeState-Req (Section 4.2.3): Provide requested data in a NetNode-Reply containing Node State and Node Data TLVs.

State-Reply (Section 4.2.4): If the message contains Node State TLVs that are more recent than local state (higher update number or we lack the node data altogether), if the message also contains corresponding Node Data TLVs, update local state and reset Trickle. If the message is lacking Node Data TLVs for some Node State TLVs which are more recent than local state, reply with a NodeState-Req (Section 4.2.3) for the corresponding nodes.

Each node is responsible for publishing a valid set of data TLVs. When there is a change in a node's set of data TLVs, the update number MUST be incremented accordingly.

If a message containing Node State TLVs (Section 5.2.3) is received via unicast or multicast with the node's own node identifier and a higher update number than current local value, or the same update number and different hash, there is an error somewhere. A recommended default way to handle this is to attempt to assert local state by increasing the local update number to a value higher than that received and republish node data using the same node identifier. If this happens more than 3 times in 60 seconds and the local node identifier is not globally unique, there may be more than one router with the same node identifier on the network. If HNCP security is not enabled, a new node identifier SHOULD be generated and node data

republished accordingly. If HNCP security is enabled, this is event is highly unlikely to occur as collision of identifier hashes for public keys is highly unlikely.

In all cases, if node data for any node changes, all Trickle instances MUST be considered inconsistent (I=Imin + timer reset).

4.4. Adding and Removing Neighbors

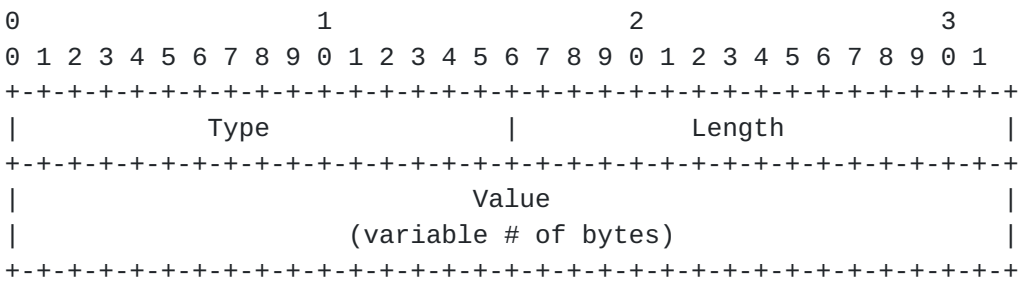
Whenever multicast message or unicast reply is received on a link from another node, the node should be added as Neighbor TLV (Section 5.2.6) for current node. If nothing (for example - no router advertisements, no HNCP traffic) is received from that neighbor in Imax seconds and the neighbor is not in neighbor discovery cache (and L2 indication of presence is available), at least 3 attempts to ping it with request network state message (Section 4.2.2) SHOULD be sent with increasing timeouts (e.g. 1, 2, 4 seconds). If even after suitable period after the last message nothing is received, the Neighbor TLV MUST be removed so that there are no dangling neighbors. As an alternative, if there is a layer 2 unreachability notification of some sort available for either whole link or for individual neighbor, it MAY be used to immediately trigger removal of corresponding Neighbor TLV(s).

4.5. Purging Unreachable Nodes

Nodes should be purged when unreachable. When node data has changed, the neighbor graph SHOULD be traversed for each node following the Neighbor TLVs, purging nodes that were found entirely unreachable (not traversed).

5. Type-Length-Value objects

Every TLV is encoded as 2 octet type, followed by 2 octet length (of the whole TLV, including header; 4 means no value whatsoever), and then the value itself (if any). The actual length of TLV MUST be always divisible by 4; if the length of the value is not, zeroed padding bytes MUST be inserted at the end of TLV. The padding bytes MUST NOT be included in the length field.



Encoding of type=123 (0x7b) TLV with value 'x' (120 = 0x78): 007B 0005 7800 0000

Notation:

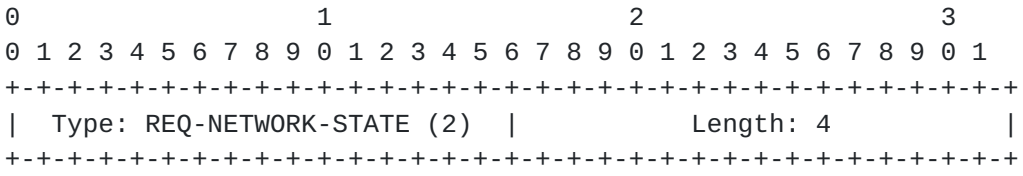
.. = octet string concatenation operation

H(x) = MD5 hash of x

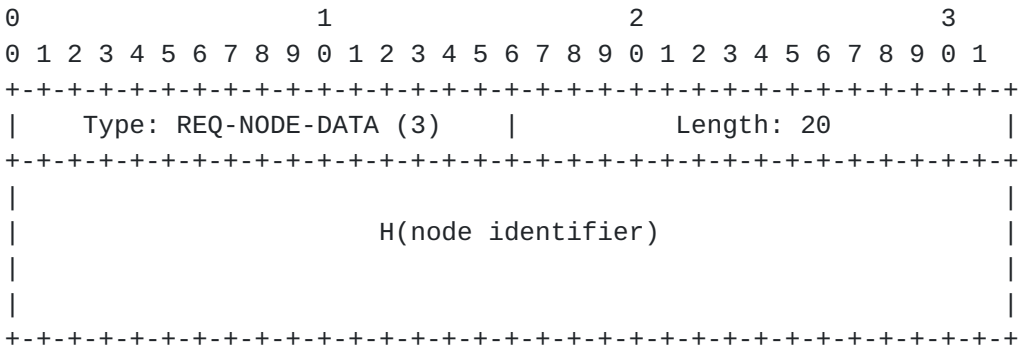
H-64(x) = H(x) truncated by taking just first 64 bits of the result.

5.1. Request TLVs (for use within unicast requests)

5.1.1. Request Network State TLV

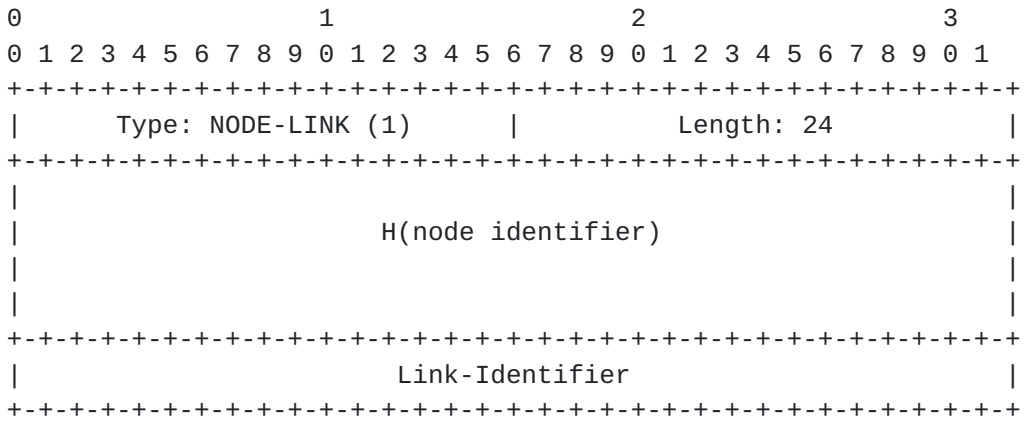


5.1.2. Request Node Data TLV

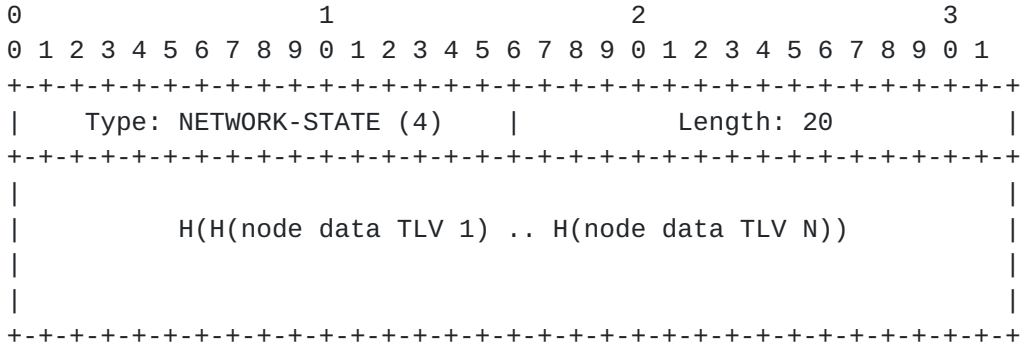


5.2. Data TLVs (for use in both multi- and unicast data)

5.2.1. Node Link TLV

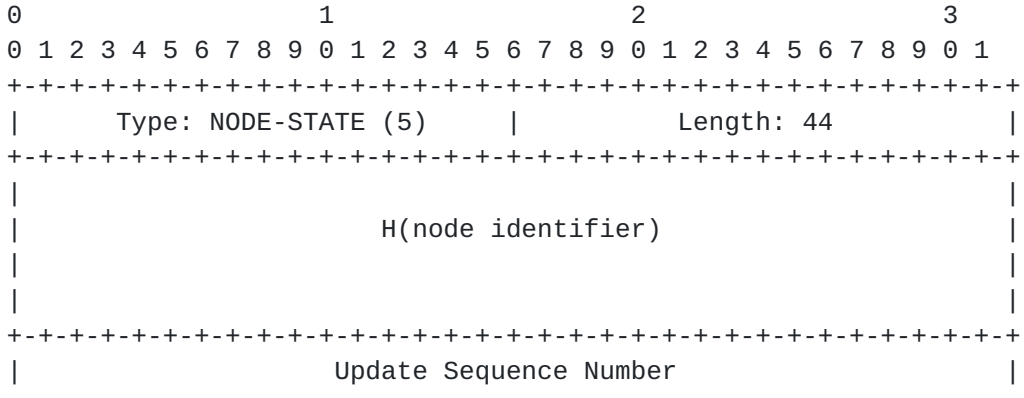


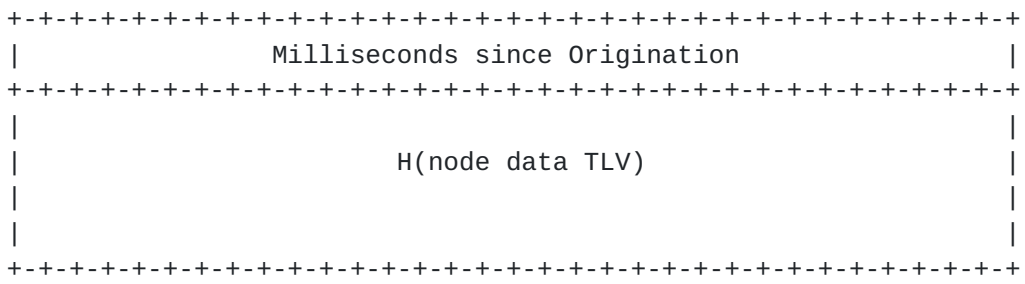
5.2.2. Network State TLV



The Node Data TLVs are ordered for hashing by octet comparison of the corresponding node identifier hashes in ascending order.

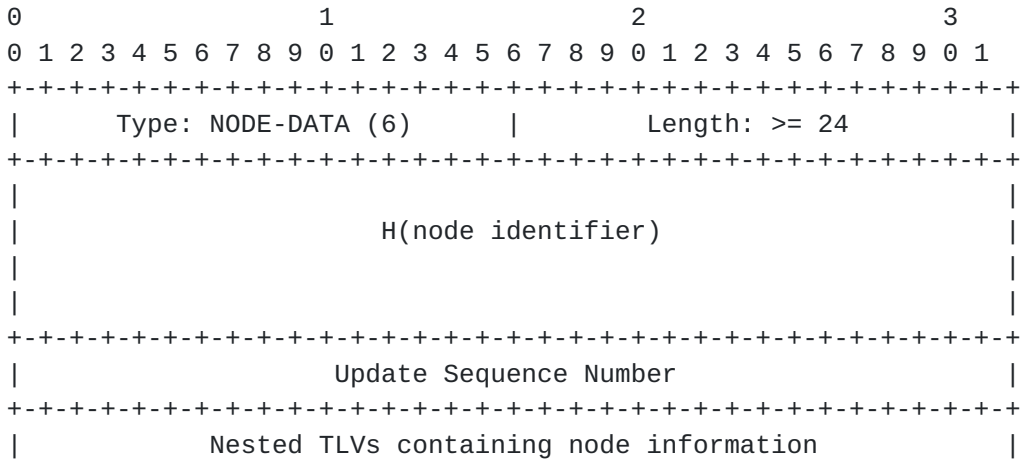
5.2.3. Node State TLV





The whole network should have roughly the same idea about the time since origination, i.e. even the originating router should increment the time whenever it needs to send a new Node State TLV regarding itself without changing the corresponding Node Data TLV. This age value is not included within the Node Data TLV, however, as that is immutable and potentially signed by the originating node at the time of origination.

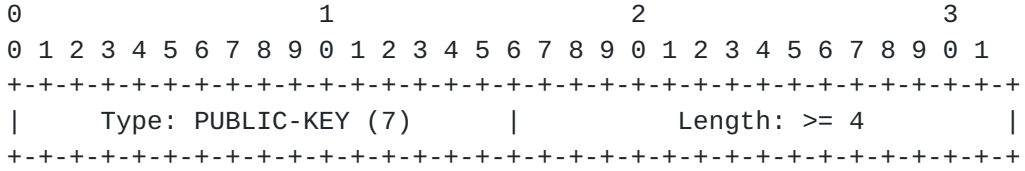
5.2.4. Node Data TLV



The Node Public Key TLV (Section 5.2.5) SHOULD be always included if signatures are ever used.

If signatures are in use, the Node Data TLV SHOULD also contain the originator's own Signature TLV (Section 5.4.2).

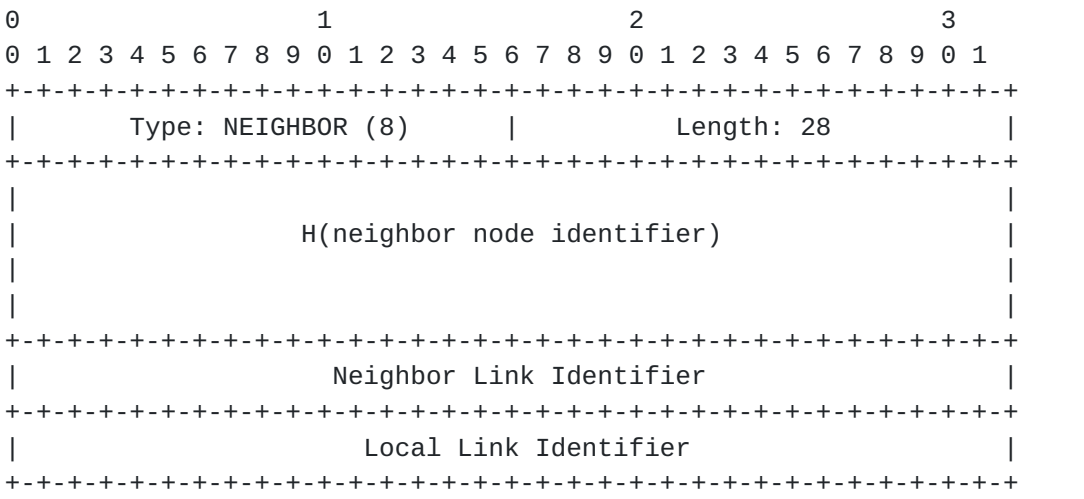
5.2.5. Node Public Key TLV (within Node Data TLV)



```
|              Public Key (raw node identifier)                |
```

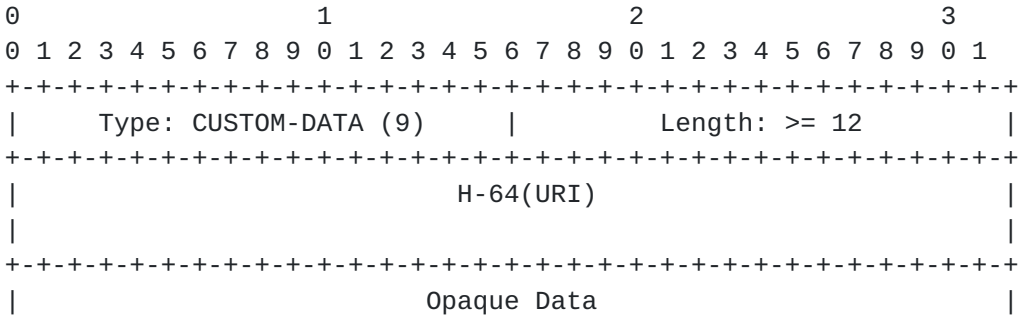
Public key data for the node. Only relevant if signatures are used. Can be used to verify that H(node identifier) equals public key, and that the Signature TLVs are signed by appropriate public keys.

5.2.6. Neighbor TLV (within Node Data TLV)



This TLV indicates that the node in question vouches that the specified neighbor is reachable by it on the local link id given. This reachability may be unidirectional (if no unicast exchanges have been performed with the neighbor). The presence of this TLV at least guarantees that the node publishing it has received traffic from the neighbor recently. For guaranteed bidirectional reachability, existence of both nodes' matching Neighbor TLVs should be checked.

5.3. Custom TLV (within/without Node Data TLV)



This TLV can be used to contain anything; the URI used should be under control of the author of that specification. For example:

V=H-64('http://example.com/author/json-for-hncp') .. '{"cool": "json extension!"}'

or

V=H-64('mailto:author@example.com') .. '{"cool": "json extension!"}'

5.4. Authentication TLVs

5.4.1. Certificate-related TLVs

TBD; should be probably some sort of certificate ID to be used in a lookup at most, as raw certificates will overflow easily IPv6 minimum MTU.

5.4.2. Signature TLV

TLV with T=0xFFFF, V=(TBD) public key algorithm based signature of all TLVs within current scope as well as the parent TLV header, if any. The assumed signature key is private key matching the public key of the the originator of node link TLV (if signature TLV is within main body of message), or that of the originator of the node data TLV (if signature TLV is within Node Data TLV)..

6. Border Discovery and Prefix Assignment

Using Default Border Definition [I-D.kline-homenet-default-perimeter] as a basis, this section defines border discovery algorithm specifics derived from the edge router interactions described in the Basic Requirements for IPv6 Customer Edge Routers [RFC7084]. The algorithm is designed to work for both IPv4 and IPv6 (single or dual-stack).

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

> A homenet router running a DHCP-client on a homenet-interface MUST include a DHCP User-Class consisting of the ASCII-String "HOMENET".
>
> A homenet router running a DHCP-server on a homenet-interface MUST ignore or reject DHCP-Requests containing a DHCP User-Class consisting of the ASCII-String "HOMENET".

An interface MUST be considered external if at least one of the following conditions is satisfied:

1. The interface has a fixed category classifying it as external.

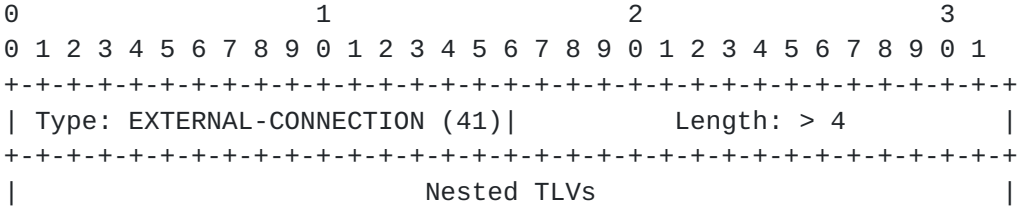
2. A delegated prefix could be acquired by running a DHCPv6-client on the interface.

3. An IPv4-address could be acquired by running a DHCP-client on the interface.

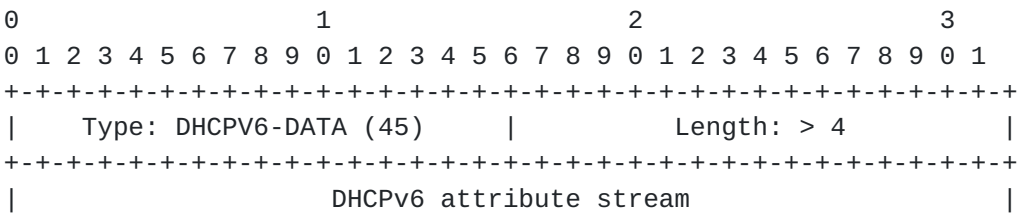
4. HNCP security is enabled and there are routers on the interface which could not be authenticated.

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 DHCP 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 possible default) in 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. Once a router detects an interface to be external it MUST stop any previously enabled internal forwarding. In addition it SHOULD announce the acquired information for use in the homenet as described in later sections of this draft if the interface appears to be connected to an external network.

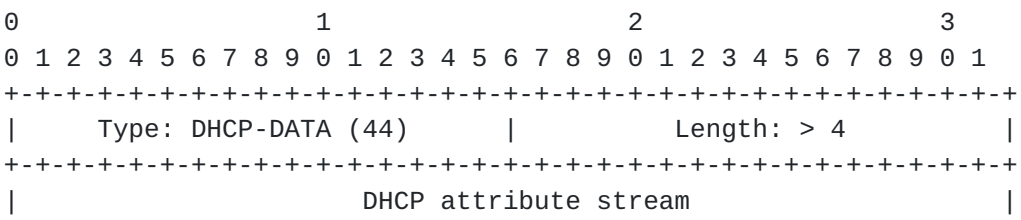
To distribute an external connection in the homenet an edge router announces one or more delegated prefixes and associated DHCP(v6)-encoded auxiliary information like recursive DNS-servers. Each external connection is announced using one container-TLV as follows:



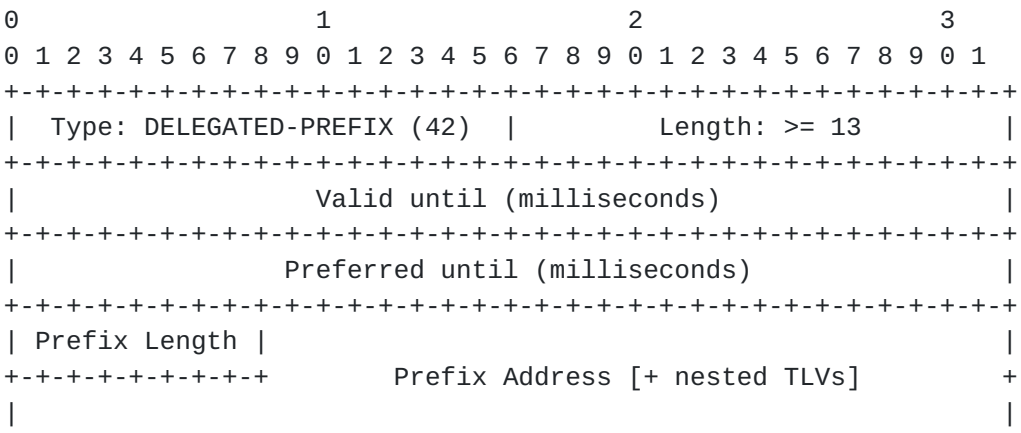
Auxiliary connectivity information is encoded as a stream of DHCPv6-attributes or DHCP-attributes placed inside a TLV of type EXTERNAL-CONNECTION or DELEGATED-PREFIX (for IPv6 prefix-specific information). There MUST NOT be more than one instance of this TLV inside a container and the order of the DHCP(v6)-attributes contained within it MUST be preserved as long as the information contained does not change. The TLVs are encoded as follows:



and



Each delegated prefix is encoded using one TLV inside an EXTERNAL-CONNECTION TLV. For external IPv4 connections the prefix is encoded in the form of an IPv4-mapped address [RFC4291] and is usually from a private address range [RFC1597]. The related TLV is defined as follows.



Valid until is the time in milliseconds the delegated prefix is valid. The value is relative to the point in time the TLV is first announced.

Preferred until is the time in milliseconds the delegated prefix is preferred. The value is relative to the point in time the TLV is first announced.

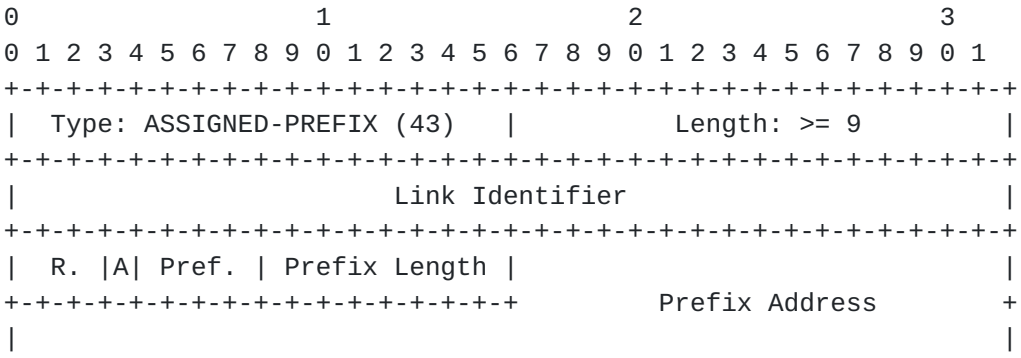
Prefix length specifies the number of significant bits in the prefix.

Prefix address is of variable length and contains the significant bits of the prefix padded with zeroes up to the next byte boundary.

Nested TLVs might contain prefix-specific information like DHCPv6-options.

In order for routers to use the distributed information, prefixes and addresses have to be assigned to the interior links of the homenet. A router MUST therefore implement the algorithm defined in Prefix and Address Assignment in a Home Network [I-D.pfister-homenet-prefix-assignment]. In order to announce the assigned prefixes the following TLVs are defined.

Each assigned prefix is given to an interior link and is encoded using one TLVs. Assigned IPv4 prefixes are stored as mapped IPv4-addresses. The TLV is defined as follows:



Link Identifier is the local HNCP identifier of the link the prefix is assigned to.

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

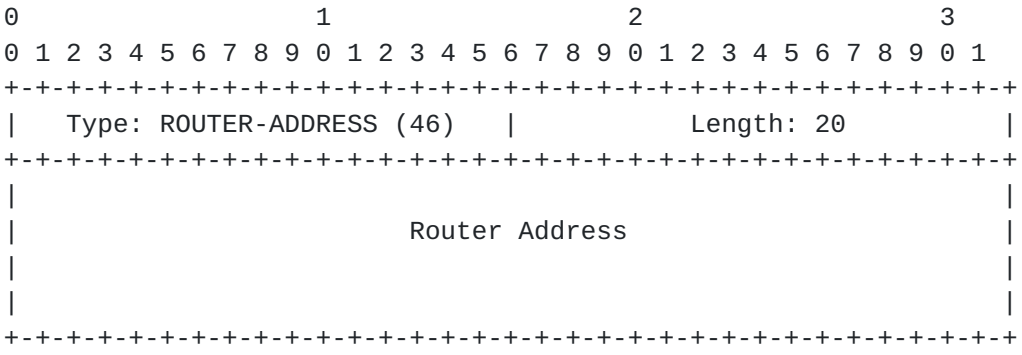
A is the authoritative flag which indicates that an assigment is enforced and ignores usual collision detection rules.

Pref. describes the preference of the assignment and can be used to differentiate the importance of a given assignment over others.

Prefix length specifies the number of significant bits in the prefix.

Prefix address is of variable length and contains the significant bits of the prefix padded with zeroes up to the next byte boundary.

In some cases (e.g. IPv4) the set of addresses is very limited and stateless mechanisms are not really suitable for address assignment. Therefore HNCP can manage router address in these cases by itself. Each router assigning an address to one of its interfaces announces one TLV of the following kind:

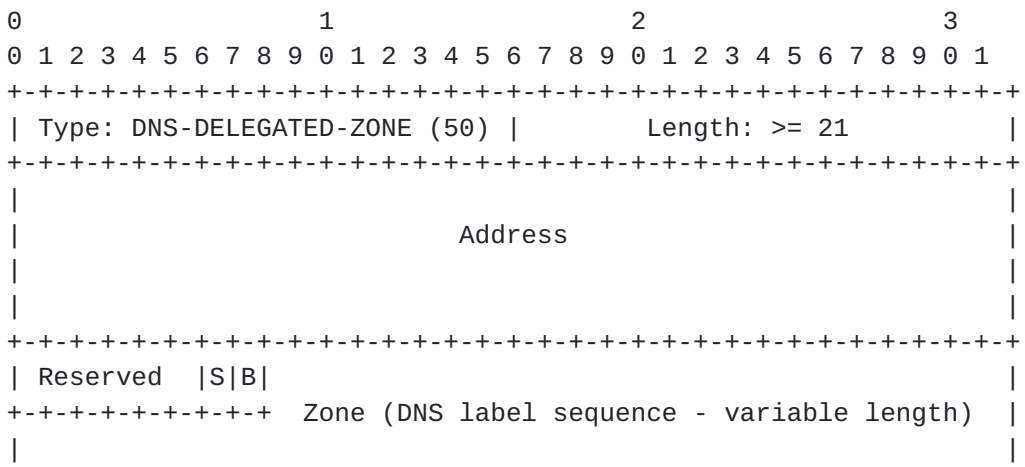


Router Address is the address assigned to one of the router interfaces.

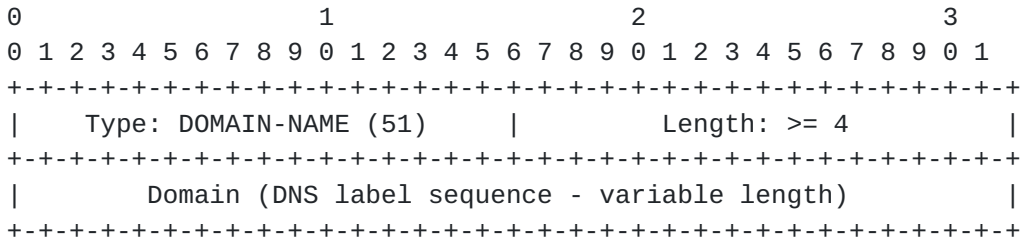
7. DNS-based Service Discovery

Service discovery is generally limited to a local link. [I-D.stenberg-homenet-dnssd-hybrid-proxy-network-zeroconf] defines a mechanism to automatically extended DNS-based service discovery across multiple links within the home automatically. Of the three TLVs, the DNS Delegated Zone TLV MUST be supported, and the remaining two SHOULD be.

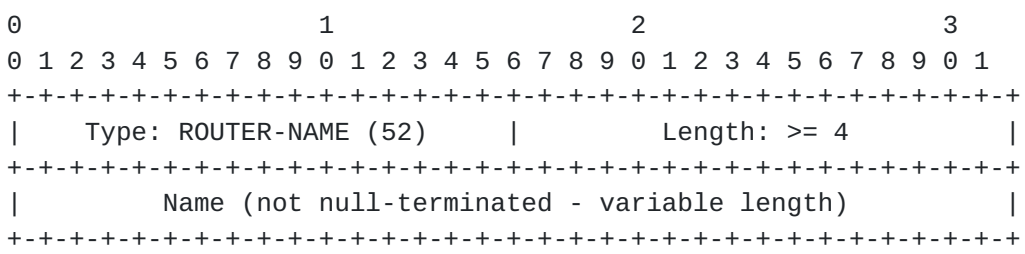
7.1. DNS Delegated Zone TLV



7.2. Domain Name TLV



7.3. Router Name TLV



8. Routing support

8.1. Protocol Requirements

In order to be advertised for use within the Homenet, a routing protocol MUST:

- Comply with Requirements and Use Cases for Source/Destination Routing [I-D.baker-rtgwg-src-dst-routing-use-cases].

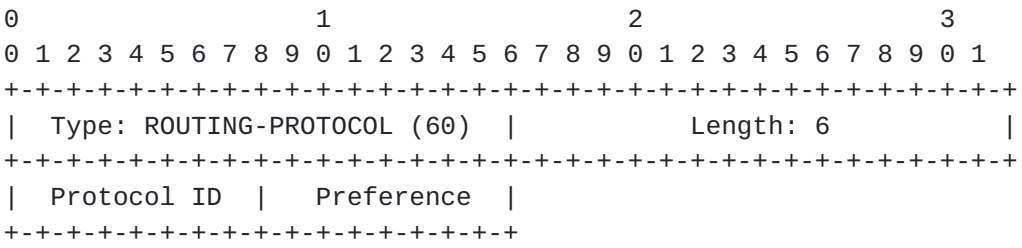
Be configured with suitable defaults or have an autoconfiguration mechanism (e.g. [I-D.acee-ospf-ospfv3-autoconfig]) such that it will run in a Homenet without requiring specific configuration from the Home user.

A router MUST NOT announce that it supports a certain routing protocol if its implementation of the routing protocol does not meet these requirements, e.g. it does not implement extensions that are necessary for compliance.

8.2. Announcement

Each router SHOULD announce all routing protocols that it is capable of supporting in the Homenet. It SHOULD assign a preference value for each protocol that indicates its desire to use said protocol over other protocols it supports and SHOULD make these values configurable.

Each router includes one HNCP TLV of type ROUTING-PROTOCOL for every such routing protocol. This TLV is defined as follows:



Protocol ID is one of:

0 = reserved

1 = Babel (dual-stack)

2 = OSPFv3 (dual-stack)

3 = IS-IS (dual-stack)

4 = RIP (dual-stack)

Preference is a value from 0 to 255. If a router is neutral about a routing protocol it SHOULD use the value 128, otherwise a lower value indicating lower preference or a higher value indicating higher preference respectively.

8.3. Protocol Selection

When HNCP detects that a router has joined or left the Homenet it MUST examine all advertised routing protocols and preference values from all routers in the Homenet in order to find the one routing protocol which:

1. Is understood by all routers in the homenet

2. Has the highest preference value among all routers (calculated as sum of preference values)

3. Has the highest protocol ID among those with the highest preference

If the router protocol selection results in the need to change from one routing protocol to another on the homenet, the router MUST stop the previously running protocol, remove associated routes, and start the new protocol in a graceful manner. If there is no common routing protocol available among all Homenet routers, routers MUST utilize the Fallback Mechanism (Section 8.4).

8.4. Fallback Mechanism

In cases where there is no commonly supported routing protocol available the following fallback algorithm is run to setup routing and preserve interoperability among the homenet. While not intended to replace a routing protocol, this mechanism provides a valid - but not necessarily optimal - routing topology. This algorithm uses the node and neighbor state already synchronized by HNCP, and therefore does not require any additional protocol message exchange.

1. Interpret the neighbour information received via HNCP as a graph of connected routers.

2. Use breadth-first traversal to determine the next-hop and hop-count in the path to each router in the homenet:

   Start the traversal with the immediate neighbours of the router running the algorithm.

   Always visit the immediate neighbours of a router in ascending order of their router ID.

   Never visit a router more often than once.

3. For each delegated prefix P of any router R in the homenet: Create a default route via the next-hop for R acquired in #2.

Each such route MUST be source-restricted to only apply to traffic with a source address within P and its metric MUST reflect the hop-count to R.

4. For each assigned prefix A of a router R: Create a route to A via the next-hop for R acquired in #2. Each such route MUST NOT be source-restricted.

5. For the first router R visited in the traversal annuncing an IPv4-uplink: Create a default IPv4-route via the next-hop for R acquired in #2.

6. For each assigned IPv4-prefix A of a router R: Create an IPv4-route to A via the next-hop for R acquired in #2.

9. Security Considerations

General security issues for Home Networks are discussed at length in [I-D.ietf-homenet-arch]. The protocols used to setup IP in home networks today have very little security enabled within the control protocol itself. For example, DHCP has defined [RFC3118] to authenticate DHCP 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. HNCP aims to make security as easy as possible for the implementer by including built-in capabilities for authentication of node data being exchanged as well as the protocol messages themselves, but it is ultimately up to the shipping system to take advantage of the protocol constructs defined.

HNCP is designed to integrate with trusted bootstrapping [I-D.behringer-homenet-trust-bootstrap] including the ability to authenticate messages between nodes. This authentication can be used to securely define a border as well as protect against malicious attacks and spoofing attempts from inside or outside the border.

HNCP itself sends messages as (possibly authenticated) clear text which is as secure, or insecure, as the security of the link below as discussed in [I-D.kline-homenet-default-perimeter]. When no unique public key is available, a hardware fingerprint or equivalent to identify routers must be available for use by HNCP.

As HNCP messages are sent over UDP/IP, IPsec may be used for confidentiality or additional message authenitation. However, this requires manually keyed IPsec per-port granularity for port IANA-UDP-PORT UDP traffic. Also, a pre-shared key has to be utilized in this case given IKE cannot be used with multicast traffic.

If no router can be trusted and additional guarantees about source of node status updates is necessary, real public and private keys should be used to create signatures and verify them in HNCP on both on per-node data TLVs as well as across the entire HNCP message. In this mode, care must be taken in rate limiting verification of invalid packets, as otherwise denial of service may occur due to exhaustion of computation resources.

As a performance optimization, instead of providing signatures for actual node data and the protocol messages themselves, it is also possible to provide signatures just for protocol messages. While this means it is no longer possible to verify the original source of the node data itself, as long as the set of routers is trusted (i.e., no router in the set has itself been hacked to provide malicious node data) then one can assume the node data is trusted because the router is trusted and the data arrived in a protected protocol message.

10. IANA Considerations

IANA should set up a registry (policy TBD) for HNCP TLV types, with following initial contents:

0: Reserved (should not happen on wire)

1: Node link

2: Request network state

3: Request node data

4: Network state

5: Node state

6: Node data

7: Node public key

8: Neighbor

9: Custom

41: External connection

42: Delegated prefix

43: Assigned prefix

44: DHCP-data

45: DHCPv6-data

46: Router-address

50: DNS Delegated Zone

51: Domain name

52: Node name

60: Routing protocol

65535: Signature

HNCP will also require allocation of a UDP port number IANA-UDP-PORT, as well as IPv6 link-local multicast address IANA-MULTICAST-ADDRESS.

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.

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

[I-D.pfister-homenet-prefix-assignment] Pfister, P., Arkko, J., and B. Paterson, "Prefix and Address Assignment in a Home Network", draft-pfister-homenet-prefix-assignment-00 (work in progress), January 2014.

[I-D.stenberg-homenet-dnssd-hybrid-proxy-network-zeroconf] Stenberg, M., "Auto-Configuration of a Network of Hybrid Unicast/Multicast DNS-Based Service Discovery Proxy Nodes ", draft-pfister-homenet-prefix-assignment-00 (work in progress), January 2014.

11.2. 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.

[RFC1597] Rekhter, Y., Moskowitz, R., Karrenberg, D., and G. de Groot, "Address Allocation for Private Internets", RFC 1597, March 1994.

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

[I-D.ietf-homenet-arch]
Chown, T., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", draft-ietf-homenet-arch-11 (work in progress), October 2013.

[I-D.troan-homenet-sadr]
Troan, O. and L. Colitti, "IPv6 Multihoming with Source Address Dependent Routing (SADR)", draft-troan-homenet-sadr-01 (work in progress), September 2013.

[I-D.behringer-homenet-trust-bootstrap]
Behringer, M., Pritikin, M., and S. Bjarnason, "Bootstrapping Trust on a Homenet", draft-behringer-homenet-trust-bootstrap-00 (work in progress), October 2012.

[I-D.baker-rtgwg-src-dst-routing-use-cases]
Baker, F., "Requirements and Use Cases for Source/ Destination Routing", draft-baker-rtgwg-src-dst-routing-use-cases-00 (work in progress), August 2013.

[I-D.kline-homenet-default-perimeter]

Kline, E., "Default Border Definition", draft-kline-homenet-default-perimeter-00 (work in progress), March 2013.

[I-D.arkko-homenet-prefix-assignment]
Arkko, J., Lindem, A., and B. Paterson, "Prefix Assignment in a Home Network", draft-arkko-homenet-prefix-assignment-04 (work in progress), May 2013.

[I-D.stenberg-homenet-dnssdext-hybrid-proxy-ospf]
Stenberg, M., "Hybrid Unicast/Multicast DNS-Based Service Discovery Auto-Configuration Using OSPFv3", draft-stenberg-homenet-dnssdext-hybrid-proxy-ospf-00 (work in progress), June 2013.

[I-D.acee-ospf-ospfv3-autoconfig]
Lindem, A. and J. Arkko, "OSPFv3 Auto-Configuration", draft-acee-ospf-ospfv3-autoconfig-03 (work in progress), July 2012.

Appendix A. Some Outstanding Issues

Should we use MD5 hashes, or EUI-64 node identifier to identify nodes?

Is there a case for non-link-local unicast? Currently explicitly stating this is link-local only protocol.

Consider if using Trickle with k=1 really pays off, as we need to do reachability checks if L2 doesn't provide them periodically in any case. Using Trickle with k=inf would remove the need for unicast reachability checks, but at cost of extra multicast traffic. On the other hand, N*(N-1)/2 unicast reachability checks when lot of routers share a link is not appealing either.

Should we use something else than MD5 as hash? It IS somewhat insecure; however signature stuff (TBD) should rely on it mainly for security in any case, and MD5 is used in a non-security role.

Appendix B. Some Obvious Questions and Answers

Q: Why not use TCP?

A: It doesn't address the node discovery problem. It also leads to N*(N-1)/2 connections when N nodes share a link, which is awkward.

Q: Why effectively build a link state routing protocol without routing?

A: It felt like a good idea at the time. It does not require periodic flooding except for very minimal Trickle-based per-link state maintenance (potentially also neighbor reachability checks if so desired).

Q: Why not multicast-only?

A: It would require defining application level fragmentation scheme. Hopefully the data amounts used will stay small so we just trust unicast UDP to handle 'big enough' packets to contain single node's TLV data. On some link layers unicast is also much more reliable than multicast, especially for large packets.

Q: Why so long IDs? Why real hash even in insecure mode?

A: Scalability of protocol isn't really affected by using real (=cryptographic) hash function.

Q: Why trust IPv6 fragmentation in unicast case? Why not do L7 fragmentation?

A: Because it will be there for a while at least. And while PMTU et al may be problems on open internet, in a home network environment UDP fragmentation should NOT be broken in the foreseeable future.

Q: Should there be nested container syntax that is actually self-describing? (i.e. type flag that indicates container, no body except sub-TLVs?)

A: Not for now, but perhaps valid design.. TBD.

Q: Why not doing (performance thing X, Y or Z)?

A: This is designed mostly to be minimal (only timers Trickle ones; everything triggered by Trickle-driven messages or local state changes). However, feel free to suggest better (even more minimal) design which works.

Appendix C. Draft source

As usual, this draft is available at https://github.com/fingon/ietf-drafts/ [3] 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 D. Acknowledgements

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

Authors' Addresses

Markus Stenberg
Helsinki 00930
Finland

Email: markus.stenberg@iki.fi

Steven Barth

Email: cyrus@openwrt.org