Project Number:	**IST-2001-32603**
Project Title:	**6NET**
CEC Deliverable Number:	**32603/HUNGARNET/DS/3.2.3/A1**
Contractual Date of Delivery to the CEC:	31 December 2002
Actual Date of Delivery to the CEC:	17 January 2003
Title of Deliverable:	DHCPv6 implementation and test report
Work package contributing to Deliverable:	WP3
Type of Deliverable*:	RP
Deliverable Security Class**:	PU
Editors:	János Mohácsi
Contributors:	Bartosz Belter, Ralph Droms, Francis Dupont, Bartosz Gajda, József Kadlecsik, Marcin Kamiński, János Mohácsi, Stig Veenas

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

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

Abstract: This document describes the needs why the DHCPv6 protocol has developed, the design and usage issues we found during the tests we performed. We found that no complete DHCPv6 is available till now. We also felt that operation of DHCPv6 in concert with DNS can be difficult, therefore a tool is under development in the project to ease the administrative burden of IPv6 administrators.

Keywords: DHCP, IPv6, Autoconfiguration, Testing, DNS, LDAP, network management

Table of Contents

1. Introduction

After many years of work, it now looks like DHCP for IPv6 is becoming stable, and will be released as an RFC soon. Despite the existence of stateless autoconfiguration for IPv6 (RFC 2462), there is still a need for DHCP. DHCP can be a complement to the stateless autoconfiguration where it can supply hosts with DNS, NTP and other configuration data. DHCP can also take care of address allocations, and replace stateless autoconfiguration completely.

The idea to use the DHCPv4 address allocation system for IPv6 where 2^64 addresses are available per link is really doubtful. Thus a strip-down “light-weight" version of DHCPv6 was developed but this is in fact a different protocol, however it is based on DHCPv6 in order to save design and implementation time.

1.1. Using DHCP together with stateless autoconfiguration

A typical host will need to configure at least IP addresses and a recursive DNS server address in order to be used. The major problem of the current stateless autoconfiguration, is that it does not supply a DNS server address. The DNS server address might be a bit more stable, but it's still a problem to find and configure the correct address. People have suggested various techniques for configuring this, DHCP being but one of them (the others included multicasting and anycasting). DHCP assessed as a good solution, since a client might also need other configuration data like domain search path, NTP servers etc. Some have claimed that DHCP is too complex, but a DHCP server in an environment with stateless autoconfiguration does not need to support IP address delegations, and does not need any per-client state. There are also other features that could be omitted in a DHCP server if necessary. Also note that even if the client has an address from stateless autoconfiguration, it might wish to request additional addresses from DHCP, some possible reasons are described in the next section.

1.2. Using DHCP instead of stateless autoconfiguration

In this case we not only wish to configure DNS etc. as described in previous section, but also IP addresses. There are several reasons one might want to do this. Stateless autoconfiguration as described in RFC 2462 creates addresses based on interface identifiers that are typically EUI-64 identifiers. On e.g. ethernet this will be created from the MAC address on the hosts ethernet interface. This means that the IPv6 address will depend on the physical ethernet interface. One might wish for a host to have a stable address independent of which ethernet interface is used though, and there are also some privacy concerns. It can also be a pain to have meaningful PTR records in the DNS for reverse lookups. DHCP can help to fulfill all of these requirements.

2. Overview of the standardisation of DHCPv6

Several years ago, the IETF took on the initiative to develop a version of DHCP for IPv6 (DHCPv6). The specification became a Dynamic Host Configuration working group (DHC WG) work item and has been under development in that working group since the initiative was started.

There are a couple of reasons for the long development and approval process for DHCPv6. While DHCPv6 is similar to DHCPv4 [RFC2131, RFC2132] in its goals and scope, all of the details of the protocol operation are different. For example, because the configuration of an interface with multiple IPv6 addresses is a fundamental feature of IPv6, DHCPv6 can manage the assignment of multiple addresses, potentially assigned over a period of time. In contrast, DHCPv4 can only assign a single address to an interface. Dhcpv6 also addresses several deficiencies in the DHCPv4 protocol, including the operation of relay agents and security.

Another reason for the long development period for DHCPv6 is that there has been some debate in the IETF about the utility and role for DHCPv6, so the specification has been tracking a moving target.

There have been many significant changes to the DHCPv6 specification in the revisions of the DHCPv6 Internet-Draft. Implementations of earlier drafts will not interoperate with the final specification as documented in draft-ietf-dhc-dhcpv6-28.txt. The last major changes occurred in revisions 24 and 25, so implementations of draft-ietf-dhc-dhcpv6-2[4-7].txt will likely not require extensive revision to become compliant with the final version of the specification.

One questions about the use of DHCPv6 is the specification of stateless address autoconfiguration. For IPv4, the primary use of DHCP is the assignment of IP addresses to hosts. A host can use stateless address autoconfiguration to determine IPv6 addresses independent of any server-based address assignment mechanism. However, a host that has used stateless address autoconfiguration may still require additional configuration information, such as a list of addresses for DNS servers. "Stateless DHCPv6", which is described in more detail in section 2.3, is used to provide these additional configuration parameters.

The remainder of this section addresses the status of the base DHCPv6 specification and several options for DHCPv6 that are documented as Internet-Drafts.

2.1. Status of the base draft

As of 17/12/2002, the IESG approved DHCPv6 as an Internet Proposed Standard. The final published specification for DHCPv6 will be based on the Internet-Draft "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" <draft-ietf-dhc-dhcpv6-28.txt>. There is an "Editor's Note", published with the announcement, that describes some small changes to the security mechanisms for DHCPv6 relay agents published in the final DHCPv6 Internet-Draft.

2.2. DHCPv6 Options

DHCPv6 uses "options" in the variable format section of a DHCPv6message. Several options, necessary for the operation of the protocol, are defined in section 22 of the DHCPv6 specification.

Several other options are in development and are described in the remainder of section 2.2.

2.2.1. IPv6 Prefix Options for DHCPv6

The "Prefix Option" is used for prefix delegation in DHCPv6. An ISP uses prefix delegation to configure a CPE device (typically a router) with one or more prefixes to use in the customer's

network. This option is currently a DHC WG work item and has also been discussed in the IPv6 WG. The latest specification for the Prefix Option is in draft-ietf-dhc-dhcpv6-opt-prefix-delegation.txt. The specification for the Prefix Option will be submitted to the IESG for approval as a Proposed Standard in January 2003.

2.2.2. DSTM Options for DHCPv6

There are three dual-stack transition mechanism (DSTM), <draft-ietf-ngtrans-dstm>, options for DHCPv6. The first two options are used to assign IPv4 addresses to a host using DSTM and to give the address of a DSTM tunnel endpoint. These options are documented in draft-ietf-dhc-dhcpv6-opt-dstm.

2.2.3. DSTM Ports Option for DHCPv6

The third option DSTM DHCPv6 specifies the ports to be used by a host for IPv4-mapped IPv6 addresses, and is documented in draft-ietf-dhc-dhcpv6-opt-dstm-ports.

The progress of all three DSTM options will be synchronized with the progress of the DSTM specification through the IETF.

2.2.4. DNS Configuration options for DHCPv6

There are two DNS configuration options documented in draft-ietf-dhc-dhcpv6-opt-dnsconfig. The first passes the IP addresses of a list of DNS servers to a host. The second option passes a list of domains to be used as a domain search list by the host.

These options have been discussed by both the DHC WG and the IPv6 WG, for use as a solution to the DNS configuration problem also addressed by "Well known site local unicast addresses for DNS resolver" <draft-ietf-ipv6-dns-discovery>. The DNS configuration options will be submitted to the IESG for approval as a Proposed Standard in January, 2003.

2.2.5. NIS/NIS+ Configuration Options for DHCPv6

There are four DHCPv6 options for NIS and NIS+ configuration, which are documented in draft-ietf-dhc-dhcpv6-opt-nisconfig. Two of the options provide a list of NIS servers and an NIS domain to a host, and two options provide a list of NIS+ servers and an NIS+ domain. The NIS configuration options will be submitted to the IESG for approval as a Proposed Standard in January 2003.

2.2.6. Time Configuration Options for DHCPv6

There are two DHCPv6 options for time configuration, which are documented in draft-ietf-dhc-dhcpv6-opt-timeconfig. One option provides a list of NTP servers to the host. The other option provides time zone information to the host. The time configuration options will be submitted to the IESG for approval as a Proposed Standard in January 2003.

2.2.7. Client Preferred Prefix option for DHCPv6

The client preferred prefix option allows a client to indicate the prefixes form which it would prefer to have its addresses assigned. This option is documented in draft-ietf-dhc-dhcpv6-opt-cliprefprefix, and is still under discussion by the DHC WG.

2.2.8. Load Balancing for DHCPv6

DHCPv6 provides for the operation of multiple servers. Under normal circumstances, all servers respond to each request for service and the requesting client picks a server for service. The

Internet-Draft "Load Balancing for DHCPv6" <draft-ietf-dhc-dhcpv6-loadb> describes a mechanism through which multiple cooperating DHCPv6 servers can determine which server should respond to a specific client. The DHCPv6 load balancing mechanism will be submitted to the IESG for approval as a Proposed Standard in January 2003.

2.3. Guide about the stateless DHCPv6 server

The DHCPv6 service of providing configuration information without address assignment is called "stateless DHCPv6", because the DHCPv6 server need not maintain any dynamic state about individual clients while providing the service. Stateless DHCPv6 requires only a subset of the DHCPv6 protocol and is significantly easier to implement and deploy. It is anticipated that stateless DHCPv6 will be the primary way in which DHCPv6 is used in IPv6 networks.

Stateless DHCPv6 may be provided through centralized DHCPv6 servers, similar to the deployment of DHCPv4 service. Because stateless DHCPv6 is a relatively simple protocol, it may be provided by a CPE router, using, for example, DNS configuration information configured by the CPE administrator or obtained through DHCPv6 form the ISP. Stateless DHCPv6 service may also be provided by DNS servers, which would respond directly to hosts with DNS configuration information.

The requirements for implementing stateless DHCPv6 are documented in draft-droms-dhcpv6-stateless-guide. This Internet-Draft lists the messages and services that must be implemented in servers and hosts to provide stateless DHCPv6 service.

3. Overview of the current DHCPv6 draft (draft version 28)

As mentioned in the previous section, DHCPv6 has been accepted for publication as an Internet Proposed Standard, based on the Internet-Draft draft-ietf-dhc-dhcpv6-28.txt. This overview is based on the Internet-Draft.

The architecture and message exchanges in DHCPv6 are similar to DHCPv4. A DHCPv6 client initiates a DHCPv6 transaction by first locating a DHCPv6 server, and then making a request for configuration information from that server. As in DHCPv4, an IPv6 address is assigned to a host with a lease, and the host can initiate a transaction with the DHCPv6 server to extend the lease on an address.

A DHCPv6 client uses a link-local address when exchanging messages with a DHCPv6 server. To avoid the requirement that a DHCPv6 server be attached to every link, DHCPv6 relay agents forward DHCPv6 messages between hosts and off-link servers. The mechanism through which relay agents forward DHCPv6 messages allows for the use of multiple relay agents between a host and a server. Relay agent options, through which a relay agent can provide additional information to the DHCPv6 server, are included as a design feature in the base DHCPv6 specification.

The address assignment mechanism in DHCPv6 allows for the assignment of multiple addresses to an interface, and allows for the dynamic assignment of additional addresses over time. Addresses are assigned to a host with a lease, a preferred lifetime and a valid lifetime. The mechanism can support renumbering through the assignment of new addresses whose lifetimes overlap existing addresses to allow for graceful transition. Addresses are grouped together for management into an "identity association", which the host and server exchange for address assignment. DHCPv6 can be used for assignment of temporary addresses [RFC3041].

Each DHCPv6 host has a "DHCP Unique Identifier" (DUID), which remains unchanged throughout the lifetime of the host. Servers use this DUID to identify hosts reliably even if the host roam between links.

Security is included in the DHCPv6 base specification. The security mechanism uses a framework similar to the security mechanism for DHCPv4 defined in RFC3118. In addition, security for messages exchanged between relay agents and servers is provided by the use of

IPsec.

A DHCPv6 server can trigger a message exchange with a host through the Reconfigure message. Security is included for the Reconfigure message to prevent intruder attacks against DHCPv6 clients.

Stateless DHCPv6 uses a two-message exchange between a client and a server. To obtain configuration information without address assignment through stateless DHCPv6, the host sends an

Information-request message. The DHCPv6 server responds with the requested configuration information. The DHCPv6 server can be configured with host-specific configuration, to allow for customized configuration of different classes of hosts. As described in section 2.3, stateless DHCPv6 service requires only a subset of the mechanism and messages of the full DHCPv6 protocol, and is easier to implement and deploy.

An ISP to delegate a prefix or prefixes to a customer can use the prefix delegation option. To use prefix delegation, the CPE initiates a DHCPv6 transaction with the ISP edge router. The ISP router selects the prefix or prefixes to be assigned to the customer, through the ISP's policy or customer provisioning process, and returns those prefixes to the CPE. The prefixes are then available for use in the customer's network. For example, the customer may be assigned a /48 prefix, which is delegated to the CPE through DHCPv6 prefix delegation. The CPE can then assign /64 prefixes from the delegated /48 prefix to links in the customer's network.

3.1. Differences between DHCP for IPv4 and IPv6

There are many differences, since DHCP IPv6 is a completely new protocol. We only list some of the more obvious differences here.

- Hosts always have a link local address that can be used in requests (in IPv4 0.0.0.0 is used as source address)
- Uses special multicast addresses for relay agents and servers
- No compatibility with BOOTP, since no BOOTP support on IPv6.
- Simplified two-message exchange for simple configuration cases
- A client can request multiple IPv6 addresses
- Client can send multiple unrelated requests to the same or different servers
- There is a reconfigure message where servers can tell clients to reconfigure. This feature is optional.

4. DHCPv6 Implementations overview

We found in general, that because of significant changes throughout the lifetime of the DHCPv6 specification, only the most recent implementations will likely be close to compatible with the final specifcation.

After some investigation we found the following implementations:

1. http://www.hycomat.co.uk/dhcp/dhcpv62809.tar.bz2
2. http://www.hycomat.co.uk/dhcp/kamedhcpv6linport.tar.gz
3. http://www.hycomat.co.uk/dhcp/dhcpv6.tgz
4. http://www.cs.ipv6.lancs.ac.uk/ftp-archive/Code/Alpha/DHCPv6/
5. ftp://ftp.kame.net/pub/kame/snap/
6. HP-UX implementation
7. Motorola experimental implementation
8. Cisco DHCPv6 in the IOS

1. dhcp62809.tar.bz2:

The dhcp62809 is based on the ISC DHCP3 release candidate 10. The work is located under dhcp6/work.linux-2.2. Only one README file. The development seems to be ceased in September 2001.

2. kamedhcpv6linport.tar.gz

This program is based on the KAME proof-of-concept DHCPv6 server/client. The server can distribute only the DNS (server address and domain name) and Time-zone information. However The implemented client can receive these information, but does not install them in the system configuration files. This implementation is based on subset the draft 15.

3. dhcpv6.tgz

This program is written by Dhawal Kumar in 1999/2000 for the INRIA IPv6 stack to test the DHCPv6 concept. It is based on the draft 15.

4. dhcpv6-1.0.tar.gz

This program was written by Yunzhou Li in 1996 (Very old). It was supported by Digital Equipment Corp. (DEC), University of New Hampshire (UNH) and National University of Singapore (NUS).

This is also a test program as it is stated in the description: "The purpose of this implementation is to verify the feasibility of DHCPv6 and Extensions for DHCPv6 protocols. Hopefully, the code can be useful for the further development of these two protocols and other research on IPv6."

The program is based on the draft 8 therefore it is extremely outdated.

5. KAME

The current KAME dhcp implementation is substantially different the earlier proof-of-concept dhcpv6 implementation. Earlier the goal was to test the best DNS configuration for IPv6. One candidate was the DHCPv6. For testing this concept earlier a very light-weight DHCPv6 was developed and eventually dropped.

The current KAME implementation is based on the draft version 26 ofDHCPv6, but does not implement the full DHCPv6 draft. According the authors:

"dhcp6c is incomplete and violates DHCPv6 protocol spec, in several aspects. In particular, it does not handle address allocation via DHCPv6. This is, however, rather intentional; the authors believe state-less address autoconfiguration is enough for IPv6 and will not implement the stateful method in the future."

It implements only prefix distribution, DNS options, rapid commit and site identifier. It is using the All_DHCP_Relay_Agents_and_Servers or the DHCP_Anycast address.

6. HPUX implementation

Available at:
`http://www.software.hp.com/cgi-bin/swdepot_parser.cgi/cgi/displayProductInfo.pl?productNumber=DHCPv6`

HPUX DHCPv6 supports the following features:

- IPv6 address allocation
- Multiple IP addresses for an interface
- Reconfiguration messages
- Relay mechanism
- Request for configuration parameters from different servers within the same domain
- DNS server address
- DNS suffix
- NTP server address
- NIS domain name
- NIS server address
- NIS+ client domain name
- NIS+ server address

- SLP DA address and its scope
- SLP service scope
- Timezone
- IPv6 address allocation
- New message types
- Multiple IP addresses for an interface
- Reconfiguration messages
- Relay mechanism
- Request for configuration parameters from different servers within

the same domain

It seems to be the most complete implementation till now.

7. Motorola implementation

This implementation is targeted to Linux to test address delegation of DHCPv6.

8. Cisco DHCPv6 in the IOS

The Cisco DHCPv6 implementation is part of IOS, and runs in Cisco routers. It is based on the -28 revision of the DHCPv6 specification (which has been accepted by the IESG as a Proposed Standard), the prefix delegation Internet-Draft <draft-ietf-dhc-dhcpv6-opt-prefix-delegation-01> and the DNS configuration options Internet Draft <draft-ietf-dhc-dhcpv6-opt-dnsconfig-02>.

The Cisco DHCPv6 client and server are specifically intended as a prefix delegation solution and do not implement the entire DHCPv6 protocol. At present, Cisco's DHCPv6 implements prefix delegation, the rapid-commit mechanism, stateless DHCPv6 ("DHCPv6-lite") and the following options:

- Client Identifier option
- Server Identifier option
- Option Request option
- Preference option
- Status Code Option
- Rapid Commit option
- Identity Association for Prefix Delegation option (IA_PD option)
- IA_PD Prefix option
- Domain Name Server option
- Domain Search List option

Each interface can be independently configured to run a DHCPv6 server or client. For example, a CPE router can be configured to use a DHCPv6 client on its interface to the link from an ISP for prefix delegation, and configured to provide DHCPv6 service on its interfaces to other links.

Prefixes can be delegated to a client through either a manually configured binding for the client or dynamically from apool of available prefixes.

A prefix delegated to a client may be used as a pool of prefixes for assignment to other interfaces. For example, if a CPE is delegated FEC0:0:1::/48, the client can assign FEC0:0:1:0::/64, FEC0:0:1:1::/64, etc., to interfaces on links inside the customer premises.

5. Test report of the DHCPv6 implementations

5.1. Kame FreeBSD

Archive:

kame-20021118-freebsd47-snap.tgz

Source:

ftp://ftp.kame.net/pub/kame/snap/

Date:

kame-snapshot: Novemver 18, 2002 dhcp6: September 24, 2002

Implemented draft:

draft-ietf-dhc-dhcpv6-26.txt (server, client)

draft-ietf-dhc-dhcpv6-24.txt (relay agent)

draft-ietf-dhc-dhcpv6exts-12.txt

5.1.1. Description

KAME project is a joint effort to create a single solid software set, especially targeted at IPv6/IPsec. The KAME kit conforms to the latest set of IPv6 specifications.

DHCPv6 is a part of the KAME project, but it is a test implementation, which is not compiled in default compilation. It is one of the most recent of all tested implementations of DHCPv6.

5.1.2. Documentation

With this implementation of DHCPv6, the following documentation is available:

- man pages

  dhcp6c(8) describes command line options, handled signals and location of configuration files.

  dhcp6s(8) describes command line options and location of configuration files.

  dhcp6relay(8) describes command line options.

  dhcp6c.conf(5) describes syntax of client configuration file

  dhcp6s.conf(5) describes syntax of server configuration file

- KAME documentation

  It contains the whole of the KAME project documentation. The **IMPLEMENTATION** file describes behaviors/implementations choices of the

KAME stack, one of its section specifies IETF drafts, which are implemented in dhcp6.

5.1.3. Functionality

According to the documentation it supports the following features:

- rapid-commit **dhcp6c.conf(5)**
- prefix-delegation request **dhcp6c.conf(5), dhcp6s(8)**
- address of DNS server **dhcp6s.conf(5)**
- site-level aggregation identifier **dhcp6c.conf(5)**
- duration lease **dhcp6s.conf(5)**

According to **dhcp6s.conf(5)** the server can only assign prefixes to known clients, because it needs to know their DUIDs (a DHCP Unique IDentifier for a DHCP participant; each DHCP client and server has exactly one DUID).

According to **dhcp6c(8)**, a client does not handle address allocation via DHCPv6. According to **dhcp6s(8)** the server neither does nor will assign IPv6 addresses.

5.1.4. Tests

5.1.4.1 Environment

Tests were performed in a separated network in PSNC and HUNGARNET separately. One host was configured as a DHCPv6 server and the other one as a client. Both hosts were running under the FreeBSD system release 4.7 for x86 platform.

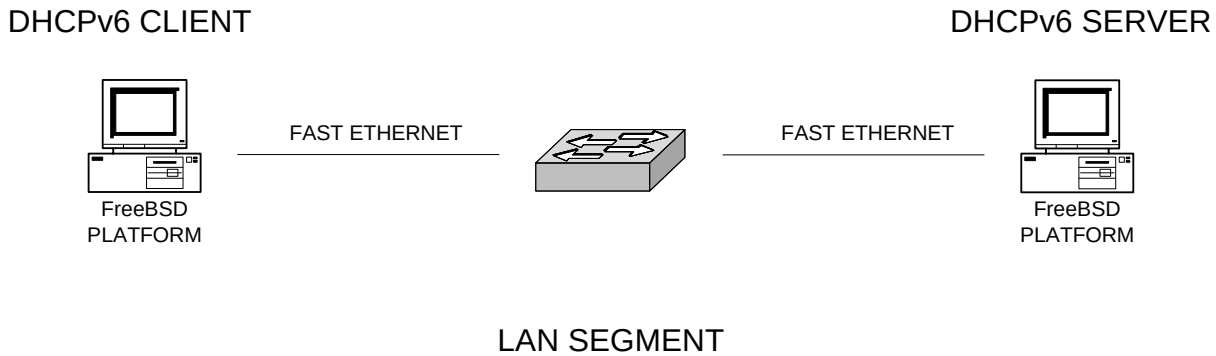

Figure 2. Scenario used for test.

5.1.4.2 Compiling the code

To compile the KAME implementation, one should extract the dhcp6 directory from archive kame-20021118-freebsd47-snap.tgz.

Next, perform the following commands:

Command line:

```
# ./configure
# ./make
```

The compilation was successful.

5.1.4.3 Test #1

Starting the server

To start the server, the following commands executed:

Command line:

```
# ./dhcp6s –c config_file –Df xl0
```

The configuration file contained the following lines:

config_file:

```
option domain-name-servers ::ffff:150.254.173.3;

host birch {
    duid 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce;
           prefix 2001:808:1::/48 infinity;
};
```

Starting the client

To start the client, the following commands executed:

Command line:

```
# ./dhcp6c –c config_file –Df xl0
```

The configuration file contained the following lines:

config_file:

```
interface xl0 {
          request prefix-delegation;
};
```

Results

Client output:

```
2002/10/28 10:45:47 get_duid: extracted an existing DUID from /etc/dhcp6c_duid:
00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:45:47 dhcp6_reset_timer: reset a timer on xl0, state=INIT, timeo=0, retrans=2383
2002/10/28 10:45:49 client6_send: a new XID (4c7070) is generated
2002/10/28 10:45:49 dhcp6_set_options: set client ID
2002/10/28 10:45:49 dhcp6_set_options: set option request
2002/10/28 10:45:49 client6_send: send solicit to ff02::1:2%xl0
2002/10/28 10:45:49 dhcp6_reset_timer: reset a timer on xl0, state=SOLICIT, timeo=0, retrans=544
2002/10/28 10:45:49 client6_recv: receive advertise from fe80::204:75ff:fec7:5a4b%xl0 on xl0
2002/10/28 10:45:49 dhcp6_get_options: get DHCP option client ID, len 14
2002/10/28 10:45:49   DUID: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:45:49 dhcp6_get_options: get DHCP option server ID, len 14
2002/10/28 10:45:49   DUID: 00:01:00:01:05:74:fc:8a:00:04:75:c7:5a:4b
2002/10/28 10:45:49 dhcp6_get_options: get DHCP option preference, len 1
2002/10/28 10:45:49 dhcp6_get_options: get DHCP option DNS, len 16
2002/10/28 10:45:49 dhcp6_get_options: get DHCP option prefix delegation, len 25
2002/10/28 10:45:49   prefix delegation option: prefix information, len 21
2002/10/28 10:45:49   prefix information: 2001:808:1::/48 duration infinity
2002/10/28 10:45:49 client6_recvadvert: server ID: 00:01:00:01:05:74:fc:8a:00:04:75:c7:5a:4b, pref=0
2002/10/28 10:45:49 client6_recvadvert: reset timer for xl0 to 0.499032
2002/10/28 10:45:50 select_server: picked a server (ID: 00:01:00:01:05:74:fc:8a:00:04:75:c7:5a:4b)
2002/10/28 10:45:50 client6_send: a new XID (5309d6) is generated
2002/10/28 10:45:50 dhcp6_set_options: set client ID
2002/10/28 10:45:50 dhcp6_set_options: set server ID
2002/10/28 10:45:50 dhcp6_set_options: set option request
2002/10/28 10:45:50 dhcp6_set_options: set prefix delegation
2002/10/28 10:45:50 client6_send: send request to ff02::1:2%xl0
2002/10/28 10:45:50 dhcp6_reset_timer: reset a timer on xl0, state=REQUEST, timeo=0, retrans=244
2002/10/28 10:45:50 client6_recv: receive reply from fe80::204:75ff:fec7:5a4b%xl0 on xl0
2002/10/28 10:45:50 dhcp6_get_options: get DHCP option client ID, len 14
2002/10/28 10:45:50   DUID: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:45:50 dhcp6_get_options: get DHCP option server ID, len 14
2002/10/28 10:45:50   DUID: 00:01:00:01:05:74:fc:8a:00:04:75:c7:5a:4b
2002/10/28 10:45:50 dhcp6_get_options: get DHCP option DNS, len 16
```

```
2002/10/28 10:45:50 dhcp6_get_options: get DHCP option prefix delegation, len 25
2002/10/28 10:45:50   prefix delegation option: prefix information, len 21
2002/10/28 10:45:50   prefix information: 2001:808:1::/48 duration infinity
2002/10/28 10:45:50 client6_recvreply: nameserver[0] ::ffff:150.254.173.3
2002/10/28 10:45:50 prefix6_add: try to add prefix 2001:808:1::/48
2002/10/28 10:45:50 dhcp6_remove_event: removing an event on xl0, state=3
2002/10/28 10:45:50 client6_recvreply: got an expected reply, sleeping.
```

Server output:

```
2002/10/28 10:46:01 configure_host: configure DUID for birch: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:46:01 get_duid: extracted an existing DUID from /etc/dhcp6s_duid:
00:01:00:01:05:74:fc:8a:00:04:75:c7:5a:4b
2002/10/28 10:46:05 server6_recv: received solicit from fe80::201:2ff:feb5:8fce%xl0
2002/10/28 10:46:05 dhcp6_get_options: get DHCP option client ID, len 14
2002/10/28 10:46:05   DUID: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:46:05 dhcp6_get_options: get DHCP option option request, len 2
2002/10/28 10:46:05   requested option: prefix delegation
2002/10/28 10:46:05 server6_react_solicit: client ID 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:46:05 server6_react_solicit: found a host configuration for birch
2002/10/28 10:46:05 dhcp6_set_options: set client ID
2002/10/28 10:46:05 dhcp6_set_options: set server ID
2002/10/28 10:46:05 dhcp6_set_options: set preference
2002/10/28 10:46:05 dhcp6_set_options: set DNS
2002/10/28 10:46:05 dhcp6_set_options: set prefix delegation
2002/10/28 10:46:05 server6_send: transmit advertise to fe80::201:2ff:feb5:8fce%xl0
2002/10/28 10:46:06 server6_recv: received request from fe80::201:2ff:feb5:8fce%xl0
2002/10/28 10:46:06 dhcp6_get_options: get DHCP option client ID, len 14
2002/10/28 10:46:06   DUID: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:46:06 dhcp6_get_options: get DHCP option server ID, len 14
2002/10/28 10:46:06   DUID: 00:01:00:01:05:74:fc:8a:00:04:75:c7:5a:4b
2002/10/28 10:46:06 dhcp6_get_options: get DHCP option option request, len 2
2002/10/28 10:46:06   requested option: prefix delegation
2002/10/28 10:46:06 dhcp6_get_options: get DHCP option prefix delegation, len 25
2002/10/28 10:46:06   prefix delegation option: prefix information, len 21
2002/10/28 10:46:06   prefix information: 2001:808:1::/48 duration infinity
2002/10/28 10:46:06 server6_react_request: found a host configuration named birch
2002/10/28 10:46:06 add_binding: add a new binding [prefix: 2001:808:1::/48, duration=-1] for
00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:46:06 dhcp6_set_options: set client ID
2002/10/28 10:46:06 dhcp6_set_options: set server ID
2002/10/28 10:46:06 dhcp6_set_options: set DNS
2002/10/28 10:46:06 dhcp6_set_options: set prefix delegation
2002/10/28 10:46:06 server6_send: transmit reply to fe80::201:2ff:feb5:8fce%xl0
```

The exchanged messages as seen through the **tcpdump** tool:

Packets:

```
10:46:05.939764 fe80::201:2ff:feb5:8fce.dhcpv6-client > ff02::1:2.dhcpv6-server: [udp sum ok] dhcp6 solicit
[hlim 1] (len 36)

                    6000 0000 0024 1101 fe80 0000 0000 0000

                    0201 02ff feb5 8fce ff02 0000 0000 0000

                    0000 0000 0001 0002 0222 0223 0024 6347

                    014c 7070 0001 000e 0001 0001 0574 fd22

                    0001 02b5 8fce 0006 0002 001e

10:46:05.940343 fe80::204:75ff:fec7:5a4b.1049 > fe80::201:2ff:feb5:8fce.dhcpv6-client: [udp sum ok] dhcp6
advert    (solicit-ID=112    pref=112    cliaddr=1:e:1:1:574:fd22:1:2b5    relayaddr=8fce:2:e:1:1:574:fc8a:4
servaddr=75c7:5a4b:7:1:0:1900:1000:0(unknown, len=0)) (len 102, hlim 64)

                    6000 0000 0066 1140 fe80 0000 0000 0000

                    0204 75ff fec7 5a4b fe80 0000 0000 0000

                    0201 02ff feb5 8fce 0419 0222 0066 feb2

                    024c 7070 0001 000e 0001 0001 0574 fd22

                    0001 02b5 8fce 0002 000e 0001 0001 0574

                    fc8a 0004 75c7 5a4b 0007 0001 0000 1900

                    1000 0000 0000 0000 0000 00ff ff96 fead

                    0300 1e00 1900 1f00 15ff ffff ff30 2001

                    0808 0001 0000 0000 0000 0000 0000

10:46:06.449590 fe80::201:2ff:feb5:8fce.dhcpv6-client > ff02::1:2.dhcpv6-server: [udp sum ok] dhcp6 request
(R           xid=0xd609           cliaddr=1:e:1:1:574:fd22:1:2b5           relayaddr=8fce:2:e:1:1:574:fc8a:4
servaddr=75c7:5a4b:6:2:1e:1e:19:1f) [hlim 1] (len 83)

                    6000 0000 0053 1101 fe80 0000 0000 0000

                    0201 02ff feb5 8fce ff02 0000 0000 0000

                    0000 0000 0001 0002 0222 0223 0053 bac1

                    0353 09d6 0001 000e 0001 0001 0574 fd22

                    0001 02b5 8fce 0002 000e 0001 0001 0574

                    fc8a 0004 75c7 5a4b 0006 0002 001e 001e

                    0019 001f 0015 ffff ffff 3020 0108 0800

                    0100 0000 0000 0000 0000 00

10:46:06.450227 fe80::204:75ff:fec7:5a4b.1049 > fe80::201:2ff:feb5:8fce.dhcpv6-client: [udp sum ok] dhcp6
(len 97, hlim 64)

                    6000 0000 0061 1140 fe80 0000 0000 0000

                    0204 75ff fec7 5a4b fe80 0000 0000 0000

                    0201 02ff feb5 8fce 0419 0222 0061 a018

                    0753 09d6 0001 000e 0001 0001 0574 fd22

                    0001 02b5 8fce 0002 000e 0001 0001 0574

                    fc8a 0004 75c7 5a4b 0019 0010 0000 0000

                    0000 0000 0000 ffff 96fe ad03 001e 0019

                    001f 0015 ffff ffff 3020 0108 0800 0100

                    0000 0000 0000 0000 00
```

Test summary

The test was performed in order to check the correctness of the prefix delegation mechanism. Message exchange is correct, but prefix is not assigned to interface on client side. Client does not use DHCPv6 information (DNS server address etc.) for its configuration.

5.1.4.4 Test #2

Starting the server

To start the server, the following commands executed:

Command line:

```
# ./dhcp6s –c config_file –Df xl0
```

The configuration file contained the following lines:

config_file:

```
option domain-name-servers ::ffff:150.254.173.3;

host birch {
    duid 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce;
    prefix 2001:808:1::/48 infinity;
};
```

Starting the client

To start the client, the following commands executed:

Command line:

```
# ./dhcp6c –c config_file –Df xl0
```

The configuration file contained the following lines:

config_file:

```
interface xl0 {
    send rapid-commit;
    request prefix-delegation;
};
```

Results

Client output:

```
2002/10/28 10:46:36 get_duid: extracted an existing DUID from /etc/dhcp6c_duid:
00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce

2002/10/28 10:46:36 dhcp6_reset_timer: reset a timer on xl0, state=INIT, timeo=0, retrans=2383
```

```
2002/10/28 10:46:39 client6_send: a new XID (b50b6a) is generated
2002/10/28 10:46:39 dhcp6_set_options: set client ID
2002/10/28 10:46:39 dhcp6_set_options: set rapid commit
2002/10/28 10:46:39 dhcp6_set_options: set option request
2002/10/28 10:46:39 client6_send: send solicit to ff02::1:2%xl0
2002/10/28 10:46:39 dhcp6_reset_timer: reset a timer on xl0, state=SOLICIT, timeo=0, retrans=544
2002/10/28 10:46:39 client6_recv: receive advertise from fe80::204:75ff:fec7:5a4b%xl0 on xl0
2002/10/28 10:46:39 dhcp6_get_options: get DHCP option client ID, len 14
2002/10/28 10:46:39   DUID: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:46:39 dhcp6_get_options: get DHCP option server ID, len 14
2002/10/28 10:46:39   DUID: 00:01:00:01:05:74:fc:8a:00:04:75:c7:5a:4b
2002/10/28 10:46:39 dhcp6_get_options: get DHCP option preference, len 1
2002/10/28 10:46:39 dhcp6_get_options: get DHCP option DNS, len 16
2002/10/28 10:46:39 dhcp6_get_options: get DHCP option prefix delegation, len 25
2002/10/28 10:46:39   prefix delegation option: prefix information, len 21
2002/10/28 10:46:39   prefix information: 2001:808:1::/48 duration infinity
2002/10/28 10:46:39 client6_recvadvert: unexpected advertise

2002/10/28 10:46:39 dhcp6_set_options: set client ID
2002/10/28 10:46:39 dhcp6_set_options: set rapid commit
2002/10/28 10:46:39 dhcp6_set_options: set option request
2002/10/28 10:46:39 client6_send: send solicit to ff02::1:2%xl0
2002/10/28 10:46:39 dhcp6_reset_timer: reset a timer on xl0, state=SOLICIT, timeo=1, retrans=1075
2002/10/28 10:46:39 client6_recv: receive advertise from fe80::204:75ff:fec7:5a4b%xl0 on xl0
2002/10/28 10:46:39 dhcp6_get_options: get DHCP option client ID, len 14
2002/10/28 10:46:39   DUID: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:46:39 dhcp6_get_options: get DHCP option server ID, len 14
2002/10/28 10:46:39   DUID: 00:01:00:01:05:74:fc:8a:00:04:75:c7:5a:4b
2002/10/28 10:46:39 dhcp6_get_options: get DHCP option preference, len 1
2002/10/28 10:46:39 dhcp6_get_options: get DHCP option DNS, len 16
2002/10/28 10:46:39 dhcp6_get_options: get DHCP option prefix delegation, len 25
2002/10/28 10:46:39   prefix delegation option: prefix information, len 21
2002/10/28 10:46:39   prefix information: 2001:808:1::/48 duration infinity
2002/10/28 10:46:39 client6_recvadvert: unexpected advertise
```

Server output:

```
2002/10/28 10:46:46 configure_host: configure DUID for birch: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:46:46 get_duid: extracted an existing DUID from /etc/dhcp6s_duid:
00:01:00:01:05:74:fc:8a:00:04:75:c7:5a:4b

2002/10/28 10:46:55 server6_recv: received solicit from fe80::201:2ff:feb5:8fce%xl0
2002/10/28 10:46:55 dhcp6_get_options: get DHCP option client ID, len 14
2002/10/28 10:46:55   DUID: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:46:55 dhcp6_get_options: get DHCP option rapid commit, len 0
2002/10/28 10:46:55 dhcp6_get_options: get DHCP option option request, len 2
2002/10/28 10:46:55   requested option: prefix delegation
2002/10/28 10:46:55 server6_react_solicit: client ID 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:46:55 server6_react_solicit: found a host configuration for birch
```

```
2002/10/28 10:46:55 dhcp6_set_options: set client ID
2002/10/28 10:46:55 dhcp6_set_options: set server ID
2002/10/28 10:46:55 dhcp6_set_options: set preference
2002/10/28 10:46:55 dhcp6_set_options: set DNS
2002/10/28 10:46:55 dhcp6_set_options: set prefix delegation
2002/10/28 10:46:55 server6_send: transmit advertise to fe80::201:2ff:feb5:8fce%xl0
```

The exchanged messages as seen through the **tcpdump** tool:

Packets:

```
10:46:59.120567 fe80::201:2ff:feb5:8fce.dhcpv6-client > ff02::1:2.dhcpv6-server: [udp sum ok] dhcp6 solicit
[hlim 1] (len 40)
                    6000 0000 0028 1101 fe80 0000 0000 0000
                    0201 02ff feb5 8fce ff02 0000 0000 0000
                    0000 0000 0001 0002 0222 0223 0028 c7ce
                    01b5 0b6a 0001 000e 0001 0001 0574 fd22
                    0001 02b5 8fce 000e 0000 0006 0002 001e
10:46:59.121096 fe80::204:75ff:fec7:5a4b.1055 > fe80::201:2ff:feb5:8fce.dhcpv6-client: [udp sum ok] dhcp6
advert    (solicit-ID=267    pref=106    cliaddr=1:e:1:1:574:fd22:1:2b5    relayaddr=8fce:2:e:1:1:574:fc8a:4
servaddr=75c7:5a4b:7:1:0:1900:1000:0(unknown, len=0)) (len 102, hlim 64)
                    6000 0000 0066 1140 fe80 0000 0000 0000
                    0204 75ff fec7 5a4b fe80 0000 0000 0000
                    0201 02ff feb5 8fce 041f 0222 0066 634a
                    02b5 0b6a 0001 000e 0001 0001 0574 fd22
                    0001 02b5 8fce 0002 000e 0001 0001 0574
                    fc8a 0004 75c7 5a4b 0007 0001 0000 1900
                    1000 0000 0000 0000 0000 00ff ff96 fead
                    0300 1e00 1900 1f00 15ff ffff ff30 2001
                    0808 0001 0000 0000 0000 0000 0000
```

Test summary

The test was performed in order to check the correctness of the prefix delegation with the rapid commit option. Message exchange is correct, but client shows message: 'unexpected advertisement'. Client does not assign prefix to interface.

5.1.4.5 Test #3

Starting the server

To start the server, the following commands executed:

Command line:

```
# ./dhcp6s –c config_file –Df xl0
```

The configuration file contained the following lines:

config_file:

```
option domain-name-servers ::ffff:150.254.173.3;

host birch {
    duid 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce;
    prefix 2001:808:1::/48 infinity;
};
```

Starting the client

To start the client, the following command executed:

Command line:

```
# ./dhcp6c –c config_file –Df xl0
```

The configuration file was:

config_file:

```
interface xl0 {
    information-only;
};
```

Results

Client output:

```
2002/10/28 10:47:12 <3>[interface] (9)
2002/10/28 10:47:12 <5>[xl0] (3)
2002/10/28 10:47:12 <3>begin of closure [{] (1)
2002/10/28 10:47:12 <3>[information-only] (16)
2002/10/28 10:47:12 <3>end of sentence [;] (1)
2002/10/28 10:47:12 <3>end of closure [}] (1)
2002/10/28 10:47:12 <3>end of sentence [;] (1)
2002/10/28 10:47:12 get_duid: extracted an existing DUID from /etc/dhcp6c_duid:
00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:47:12 dhcp6_reset_timer: reset a timer on xl0, state=INIT, timeo=0, retrans=2383
2002/10/28 10:47:15 client6_send: a new XID (d11e94) is generated
2002/10/28 10:47:15 dhcp6_set_options: set client ID
2002/10/28 10:47:15 client6_send: send information request to ff02::1:2%xl0
2002/10/28 10:47:15 dhcp6_reset_timer: reset a timer on xl0, state=INFOREQ, timeo=0, retrans=494
2002/10/28 10:47:15 client6_recv: receive reply from fe80::204:75ff:fec7:5a4b%xl0 on xl0
2002/10/28 10:47:15 dhcp6_get_options: get DHCP option client ID, len 14
2002/10/28 10:47:15   DUID: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:47:15 dhcp6_get_options: get DHCP option server ID, len 14
```

```
2002/10/28 10:47:15   DUID: 00:01:00:01:05:74:fc:8a:00:04:75:c7:5a:4b
2002/10/28 10:47:15 dhcp6_get_options: get DHCP option DNS, len 16
2002/10/28 10:47:15 client6_recvreply: nameserver[0] ::ffff:150.254.173.3
2002/10/28 10:47:15 dhcp6_remove_event: removing an event on xl0, state=2
2002/10/28 10:47:15 client6_recvreply: got an expected reply, sleeping.
```

Server output:

```
2002/10/28 10:47:27 configure_host: configure DUID for birch: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:47:27 get_duid: extracted an existing DUID from /etc/dhcp6s_duid: 00:01:00:01:05:74:fc:8a:00:04:75:c7:5a:4b
2002/10/28 10:47:31 server6_recv: received information request from fe80::201:2ff:feb5:8fce%xl0
2002/10/28 10:47:31 dhcp6_get_options: get DHCP option client ID, len 14
2002/10/28 10:47:31   DUID: 00:01:00:01:05:74:fd:22:00:01:02:b5:8f:ce
2002/10/28 10:47:31 dhcp6_set_options: set client ID
2002/10/28 10:47:31 dhcp6_set_options: set server ID
2002/10/28 10:47:31 dhcp6_set_options: set DNS
2002/10/28 10:47:31 server6_send: transmit reply to fe80::201:2ff:feb5:8fce%xl0
```

The exchanged messages as seen through the **tcpdump** tool:

Packets:

```
10:47:31.475450 fe80::201:2ff:feb5:8fce.dhcpv6-client > ff02::1:2.dhcpv6-server: [udp sum ok] dhcp6 [hlim 1] (len 30)
                    6000 0000 001e 1101 fe80 0000 0000 0000
                    0201 02ff feb5 8fce ff02 0000 0000 0000
                    0000 0000 0001 0002 0222 0223 001e aad0
                    0bd1 1e94 0001 000e 0001 0001 0574 fd22
                    0001 02b5 8fce
10:47:31.475904 fe80::204:75ff:fec7:5a4b.1065 > fe80::201:2ff:feb5:8fce.dhcpv6-client: [udp sum ok] dhcp6 (len 68, hlim 64)
                    6000 0000 0044 1140 fe80 0000 0000 0000
                    0204 75ff fec7 5a4b fe80 0000 0000 0000
                    0201 02ff feb5 8fce 0429 0222 0044 c599
                    07d1 1e94 0001 000e 0001 0001 0574 fd22
                    0001 02b5 8fce 0002 000e 0001 0001 0574
                    fc8a 0004 75c7 5a4b 0019 0010 0000 0000
                    0000 0000 0000 ffff 96fe ad03
```

Test summary

The test was performed in order to check the correctness of message passing. Messages are different from messages in the previous tests. They are not recognized as specific DHCP messages by **tcpdump(8)**. Server response correctly contains only DNS server address and its own DUID, but not the prefix.

5.2. Kame Linport

Archive:

kamedhcpv6linport.tar.gz

Source:

http://www.hycomat.co.uk/dhcp/kamedhcpv6linport.tar.gz

Date:

September 12, 1999

Implemented draft:

draft-ietf-dhc-dhcpv6-15.txt

5.2.1. Description

The KAME Project is a joint effort to create a single solid software set, especially targeted at IPv6/IPsec. The KAME kit conforms, or tries to conform, to the latest set of IPv6 specifications.

On the webpage this package is described as 'Dirty Linux Port from Kame.org'. It is taken from an old KAME snapshot and adapted to run under Linux.

5.2.2. Documentation

- man pages

  dhcp6c(8) describes command line options, handled signals and location of configuration files.

  dhcp6s(8) describes command line options and location of configuration files.

  dhcp6relay(8) describes command line options.

5.2.3. Functionality

According to the documentation it supports:

- address of DNS server
- DNS domain name
- current timezone offset
- IEEE 1003.1 information for current timezone

According to **dhcp6c(8)**, client does not handle resource allocations such as IPv6 address and does not install information it gets from the DHCPv6 server into the system such as DNS server, DNS domain name, timezone .

5.2.4. Tests

5.2.4.1 Environment

Tests were performed in a separated network in PSNC. One computer was configured as a DHCPv6 server and the other one as a client. Both computers run under LINUX system for x86 platform: server – PLD Ra 1.0 (glibc-2.2.5), client – SUSE 8.0 (glibc-2.2.5).

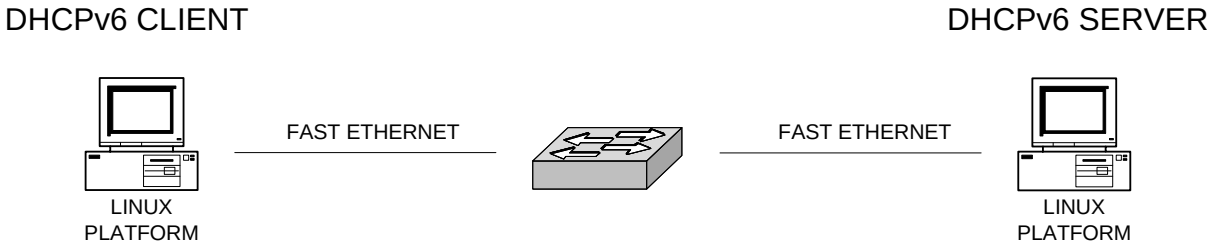

Figure 3. Scenario used for test.

5.2.4.2 Compiling the code

From kamedhcpv6linport.tar.gz the dhcp6 directory was extracted.

Next, perform the following commands:

Command line:

```
# ./configure
# ./make
```

Compilation fails on both machines because of incompatible header files, but binaries are also distributed in the archive.

5.2.4.3 Test #1

Starting the server

To start the server, the following command executed:

Command line:

```
# [root@fern dhcp6]# ./dhcp6s -Df eth0
```

Starting the client

To start the client, the following command executed:

Command line:

```
# . birch:~/dhcp/linport # ./dhcp6c -Df eth0
```

Results

Client output:

```
sendtime=1038517460 waittime=0 delaytime=1
send solicit
sendtime=1038517461 waittime=2 delaytime=3
sendtime=1038517461 waittime=2 delaytime=3
send solicit
sendtime=1038517464 waittime=2 delaytime=7
sendtime=1038517464 waittime=2 delaytime=7
no server found
```

Server output:

```
server6_recv: from , size 36
msgtype=1
react_solicit
react_solicit: invalid cliaddr:
server6_recv: from , size 36
msgtype=1
react_solicit
react_solicit: invalid cliaddr:
```

The sent messages as seen through the **tcpdump** tool:

Packets:

```
13:04:40.227330 fe80::201:2ff:feb5:8fce.32768 > ff02::1:2.547: [udp sum ok]
dhcp6 solicit (plen=0 solicit-ID=2 cliaddr=:: relayaddr=::) [hlim 1] (len
44)
```

```
6000 0000 002c 1101 fe80 0000 0000 0000
0201 02ff feb5 8fce ff02 0000 0000 0000
0000 0000 0001 0002 8000 0223 002c eb65
0100 0002 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000
```

Test summary

The test was performed in order to check the correctness of the declared functionality of this software.

Client sends a solicit message with client address equals `:: (wildcard address).` This behavior is incompatible with the current DHCPv6 draft.

Server recognizes this message as a valid solicit message with an invalid client address and denies it.

Client timeouts because of the lack of server response.

5.3. DHCPv6 by Dhawal Kumar (1999/2000)

Archive:

dhcpv6.tgz

Source:

http://www.hycomat.co.uk/dhcp/dhcpv6.tgz

Date:

1999/2000

Implemented draft:

draft-ietf-dhc-dhcpv6-15.txt

draft-ietf-dhc-dhcpv6exts-12.txt

5.3.1. Description

This is a DHCPv6 implementation made by Dhawal Kumar. This software developed for the INRIA IPv6 stack.

This distribution includes two files:

- dhcpv6.psz: Dhawal Kumar's report (in English, gzipped PostScript A4)
- dhcpv6.tgz: the software itself «as it» (gzipped tar)

5.3.2. Documentation

- http://www-ftp.lip6.fr/pub/networking/inria-ipv6/dhcpv6/dhcpv6.psz.gz

  Dhawal Kumar report. It contains the architecture for which software was designed, a short description of client, server and software features.

- *.txt files

  It contains information about the format of DHCPv6 messages, programmers guide and a short description of the software behavior.

5.3.3. Functionality

According to the documentation it supports:

- Request for any address
- Request for specific address
- Request for address release

The dhcp daemon plays a client role in tested software. It is responsible for sending and receiving packets to/from the dhcp server.

The dhcp daemon can handle multiple application clients and only one dhcp daemon per machine is allowed (see Figure 4).

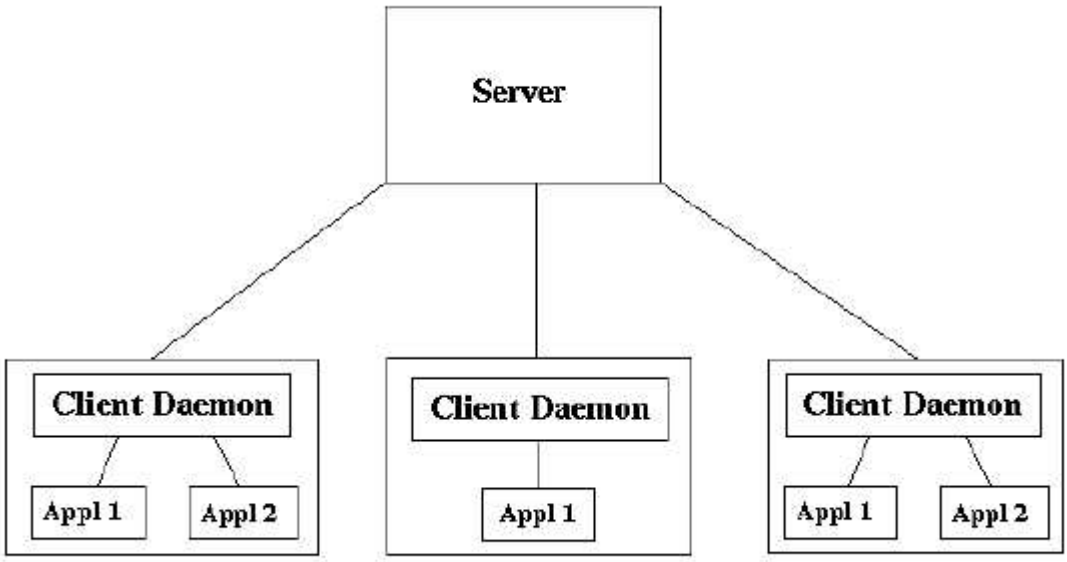

Figure 4. The Architecture of Dhawal Kumar software.

5.3.4. Tests

5.3.4.1 *Environment*

Tests were performed in a separate network at PSNC. One computer was configured as a DHCPv6 server and the other one as a client. Both computers run under FreeBSD system release 4.7 for x86 platform.

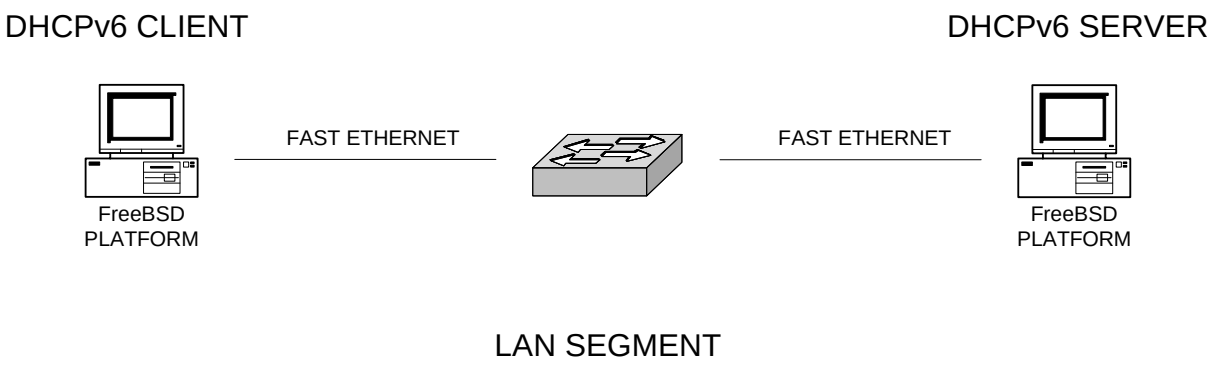

Figure 5. Scenario used for test.

5.3.4.2 Compiling the code

From dhcpv6.tgz.gz the project directory was extracted. Some changes have been made in the server's source code in order to avoid errors and warning messages during compilation:

- all appearances of IPV6_RECVPKTINFO was replaced with IPV6_PKTINFO
- all appearances of **s6_addr8** was replaced with **s6_addr**
- all appearances of **s6_addr32** was replaced with **__u6_addr.__u6_addr32**

Compilation of server source code:

Command line:

```
# ./make
```

Compilation of server source was successful.

There is no Makefile for client available and no compilation was performed, but binaries are also distributed in the archive.

5.3.4.3 Test #1

Starting the server

To start the server, the following commands executed:

Command line:

```
# ./run config_file
```

The configuration file contained the following lines:

config_file:

```
interface xl0 {
   default-preferred-lifetime 3
   default-valid-lifetime     4
   max-preferred-lifetime     5
   max-valid-lifetime         6
   scope                      3
   dns-update-A-allowed       0
   dns-update-P-allowed       1
   preference                 170

   domain-name-server {
      1111:bcde::1/64   dddd:2345:4567:5678::3/64   dddd::5/64
   }
```

```
   domain-name-suffix   man.poznan.pl

   list-of-prefixes-to-advertise {
      4ffe:305:1002:1::/64 2ffe:305:1002:1::/64
      5ffe:305:1002:1::/64
   }

   server-address             2001:808:0:1::1/64
}
```

Starting the client

To start the client, the following commands executed:

Command line:

```
# ./a.out
```

Results

Client MENU:

```
Choices: (1) init (2) any_addr (3) any_addr_with_lifetime
         (4) specific_addr (5) specific_addr_with_lifetime (6) release
         (7) DNS_UPDATE_NON_DHCP (8) ERE (9) REQUEST_WITH_CLEAR
         (10) RELEASE_DNS_RECORDS
```

Client output: (1) init

```
Enter you choice: 1
Sending init to client daemon
Sending to client daemon:
msg_type = 0 pid        = 541
request_no = 1
if_index = 1
c = 0 p = 1 all_servers = 1want_to_reconfigure = 1
Waiting for reply to init msg
Packet->
size 72-> 02 00 02 1d 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 01 08 08 00 00 00 01 00
00 00 00 00 00 00 01 00 00 aa 03 40 4f fe 03 05 10 02 00 01 40 2f fe 03 05 10 02 00 01 40 5f fe 03 05 10 02
00 01 00


Reply from client daemon


msg_type   = 2
pid        = 541
request_no = 1
```

```
Result of solicit = Success. Continuing...


-----------------------------------------
relay: ::
server: 2001:808:0:1::1
preference: 170
prefix-list: 4ffe:0305:1002:0001    2ffe:0305:1002:0001    5ffe:0305:1002:0001
```

Client output: (2) any_addr

```
Enter you choice: 2
Sending request to client daemon
Sending:< 03 00 02 22 00 00 00 02
00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 01 08 08 00 00 00 01 00 00 00 00 00 00 00 01
00 01 00 0d 00 14 00 00
62 68 6f 6f 74 2e 63 6f 6d
>
Waiting for reply to request msg
```

Server output:

```
Link local addr of [4] name <lo0> = fe80:4::1
                IPv6 addr list ffff:ffff:ffff:ffff:: ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
Link local addr of [1] name <xl0> = fe80:1::201:2ff:feb5:8fce
                IPv6 addr list 2001:808:0:1::1 ffff:ffff:ffff:ffff::


Received from fe80::204:75ff:fec7:5a4b


size 36-> 01 40 00 01 fe 80 00 01 00 00 00 00 02 04 75 ff fe c7 5a 4b 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00
Solicit.........................
c = 0 p = 1 prefix-len = 0 solicit_id = 1
size 91-> 02 00 01 aa fe 80 00 01 00 00 00 00 02 04 75 ff fe c7 5a 4b 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 20 01 08 08 00 00 00 01 00 00 00 00 00 00 00 01 20 0f 00 09 40 4f fe 03 05 10 02 00 01 20 0f 00 09
40 2f fe 03 05 10 02 00 01 20 0f 00 09 40 5f fe 03 05 10 02 00 01
.....................************.......................
addr fe80:1::204:75ff:fec7:5a4b port 546
Sending to fe80:1::204:75ff:fec7:5a4b port 546


size 91-> 02 00 01 aa fe 80 00 01 00 00 00 00 02 04 75 ff fe c7 5a 4b 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 20 01 08 08 00 00 00 01 00 00 00 00 00 00 00 01 20 0f 00 09 40 4f fe 03 05 10 02 00 01 20 0f 00 09
40 2f fe 03 05 10 02 00 01 20 0f 00 09 40 5f fe 03 05 10 02 00 01
sent_msg_to_client$$$$$$$$$$$$$$$$$$
        ...Checking expiry for 5999202 485099 at time 1038483820 698607 ...diff in micro 250058884 800000000
```

```
<1> min = 500 0
Sleeping for 500 0
```

The messages as seen through the **tcpdump** tool:

Packets:

```
12:43:55.670872 fe80::204:75ff:fec7:5a4b.dhcpv6-client > ff02::1:2.dhcpv6-server: [udp sum ok] dhcp6 solicit (P
len=0 solicit-ID=1 cliaddr=fe80:1::204:75ff:fec7:5a4b relayaddr=::) [hlim 1] (len 44)
                          6000 0000 002c 1101 fe80 0000 0000 0000
                          0204 75ff fec7 5a4b ff02 0000 0000 0000
                          0000 0000 0001 0002 0222 0223 002c 5bda
                          0140 0001 fe80 0001 0000 0000 0204 75ff
                          fec7 5a4b 0000 0000 0000 0000 0000 0000
                          0000 0000
12:43:55.672289 fe80::201:2ff:feb5:8fce.1069 > fe80::204:75ff:fec7:5a4b.dhcpv6-client: [udp sum ok] dhcp6 advert
(solicit-ID=1 pref=170 cliaddr=fe80:1::204:75ff:fec7:5a4b relayaddr=:: servaddr=2001:808:0:1::1 (Subnet Prefix,
404ffe030510020001) (Subnet Prefix, 402ffe030510020001) (Subnet Prefix, 405ffe030510020001)) (len 99, hlim 64)
                          6000 0000 0063 1140 fe80 0000 0000 0000
                          0201 02ff feb5 8fce fe80 0000 0000 0000
                          0204 75ff fec7 5a4b 042d 0222 0063 7381
                          0200 01aa fe80 0001 0000 0000 0204 75ff
                          fec7 5a4b 0000 0000 0000 0000 0000 0000
                          0000 0000 2001 0808 0000 0001 0000 0000
                          0000 0001 200f 0009 404f fe03 0510 0200
                          0120 0f00 0940 2ffe 0305 1002 0001 200f
                          0009 405f fe03 0510 0200 01
```

Test summary

The test was performed in order to check the correctness of the declared functionality of this software.

Client sends a solicit message. Server recognizes this message as a valid solicit message and replies with advertise.

Client outputs correct data extracted from the message. Then it tries to send a request, but it uses server address extracted from the message that is a global or site local. Sending fails because client has got only a link local address assigned to its interface. Its behavior is clearly no conform to the current draft.

5.4. DHCPv6 NetBSD on Alpha 64 bit

Archive:

dhcpv6-1.0.tar.gz

Source:

www.cs.ipv6.lancs.ac.uk/ftp-archive/Code/Alpha/DHCPv6/

Date:

December 08, 1996

Implemented draft:

unknown (no information in documentation)

5.4.1. Description

This is an implementation of the Dynamic Host Configuration Protocol for IPv6 on NetBSD, which is supported by Digital Equipment Corp. (DEC), University of New Hampshire (UNH) and National University of Singapore (NUS). This implementation benefits from UNH's IPv6 implementation by Quaizar Vohra.

The purpose of this implementation is to verify the feasibility of DHCPv6 and Extensions for DHCPv6 protocols.

5.4.2. Documentation

- man pages

  dhcc(8) describes the DHCPv6 client command line options and location of configuration files

  dhrc(8) describes the DHCPv6 relay agent command line options and location of configuration files

  dhsc(8) describes the DHCPv6 server command line options
- README (installation manual)
- TODO file

5.4.3. Functionality

According to man pages this package implements:

- a request for any address
- a request for specific address
- duration lease
- address release

5.4.4. Tests

5.4.4.1 Compilation

As it was pointed in the README file, the compilation process consisted in running the *make* command. Compilation under both Linux and FreeBSD systems failed because of incompatible header files.

5.4.4.2 Test summary

No tests were performed because of the lack of the Alpha machine.

5.5. Modified DHCP ISC Distribution Version 3

Archive:

dhcpv62809.tar.bz2

Source:

http://www.hycomat.co.uk/dhcp/dhcpv62809.tar.bz2

Date:

June 29, 2001

Implemented draft:

unknown (no information in the documentation, tests show ambiguous implementation comparing with standard drafts)

5.5.1. Description

This implementation is based on the original implementation of DHCP, which is supported by Internet Software Consortium. It is an attempt to adopt the existing DHCP for IPv4 solutions to new requirements (IPv6).

5.5.2. Documentation

- original documentation from ISC DHCP - IPv4 only
- http://www.hycomat.co.uk/dhcp/READMEdhcpv6.txt file with compile instructions

There is no documentation concerning DHCPv6 usage.

5.5.3. Functionality

Some changes were made in order to introduce DHCPv6, but they are not documented at all. Original DHCP server has a complicated but documented configuration file. Syntax of IPv6 changes is unknown, one small example shows how to bind dhcp-client-identifier (we do not know if this is DUID or something else) to IPv6 address (but without prefix-length).

5.5.4. Tests

5.5.4.1 *Environment*

Tests were performed at PSNC, HUNGARNET and UNINETT. At each site One computer was configured as a DHCPv6 server and the other one as a client. The software tested on LINUX system for x86 platform with different distribution. At PSNC the server – PLD Ra 1.0 (glibc-2.2.5), client – SUSE 8.0 (glibc-2.2.5). At HUNGARNET Debian linux used for both server and client.

5.5.4.2 The scenario

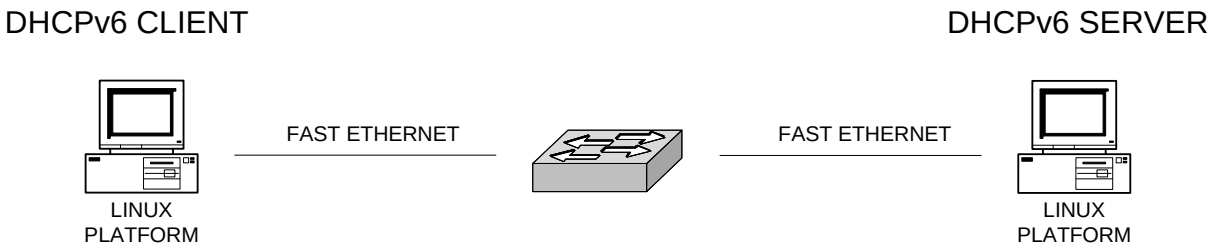

Figure 6. Scenario used for test.

5.5.4.3 Compilation

dhcpv62809.tar.bz2 tarball was unpacked.

Next, the following commands poerformed:

Command line:

```
# ./configure
# cd work.linux-2.2
# ./make
```

Compilation failed during the compilation of the relay agent, but client and server created. Compilation failure is normal and described in http://www.hycomat.co.uk/dhcp/READMEdhcpv6.txt file.

5.5.4.4 Test #1

Starting the client

To start the client, the following commands executed:

Command line:

```
# ./ birch:~/dhcp/isc/work.linux-2.2/client # ./dhclient
```

Results

```
Internet Software Consortium DHCP Client V3.0rc10
```

```
Copyright 1995-2001 Internet Software Consortium.
All rights reserved.
For info, please visit http://www.isc.org/products/DHCP

reason=PREINIT
IPV6 BIND
make option for packet type: 1
Segmentation fault
```

Test summary

Client crashed with segmentation fault on every distribution. The reason is unknown. Because of client failure another test was performed with client from other package – KAME Linport.

5.5.4.5 Test #2

Starting the server

To start the server, the following commands executed:

Command line:

```
[root@fern server]# ./dhcpd -d -f
```

The configuration file contained the following lines:

config_file:

```
# dhcpd.conf
#
# Sample configuration file for ISC dhcpd
#

# option definitions common to all supported networks...
option domain-name "ipv6.man.poznan.pl";
option domain-name-servers rose.man.poznan.pl;
ddns-update-style ad-hoc;

default-lease-time 600;
max-lease-time 7200;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
```

```
authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

# No service will be given on this subnet, but declaring it helps the
# DHCP server to understand the network topology.

subnet 0.0.0.0 netmask 255.255.255.0 {
}

subnet 150.254.170.64 netmask 255.255.255.192 {
}
host birch {
        option dhcp-client-identifier "123";
        fixed-ip6-address "2001:808:0:3::3";
}
```

Client running

This client was taken from the KAME Linport package. To start the client, the following commands executed:

Command line:

```
# . birch:~/dhcp/linport # ./dhcp6c -Df eth0
```

Results

This is a screen printout of the client:

Client output:

```
sendtime=1038517460 waittime=0 delaytime=1
send solicit
sendtime=1038517461 waittime=2 delaytime=3
sendtime=1038517461 waittime=2 delaytime=3
send solicit
sendtime=1038517464 waittime=2 delaytime=7
sendtime=1038517464 waittime=2 delaytime=7
no server found
```

This is a screen printout of the server:

Server output:

```
Internet Software Consortium DHCP Server V3.0rc10
Copyright 1995-2001 Internet Software Consortium.
All rights reserved.
For info, please visit http://www.isc.org/products/DHCP
parse_ip6_addr_or_hostname
2001:808:0:3::3
vor make_const_data
Wrote 0 deleted host decls to leases file.
Wrote 0 new dynamic host decls to leases file.
Wrote 0 leases to leases file.
IPV6 BIND
Listening on LPF/eth0/00:04:75:c7:5a:4b/150.254.170.64/26
Sending on   LPF/eth0/00:04:75:c7:5a:4b/150.254.170.64/26
GOT_ONE_6
do_packet_6: incoming packet
nur DUID received! returning
GOT_ONE_6
do_packet_6: incoming packet
nur DUID received! returning
```

These are packets passed through the **tcpdump** tool:

packets:

```
13:15:27.138970 fe80::201:2ff:feb5:8fce.32768 > ff02::1:2.547: [udp sum ok]
dhcp6 solicit (plen=0 solicit-ID=1 cliaddr=:: relayaddr=::) [hlim 1] (len
44)
                        6000 0000 002c 1101 fe80 0000 0000 0000
                        0201 02ff feb5 8fce ff02 0000 0000 0000
                        0000 0000 0001 0002 8000 0223 002c eb66
                        0100 0001 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000
13:15:30.138188 fe80::201:2ff:feb5:8fce.32768 > ff02::1:2.547: [udp sum ok]
dhcp6 solicit (plen=0 solicit-ID=2 cliaddr=:: relayaddr=::) [hlim 1] (len
44)
                        6000 0000 002c 1101 fe80 0000 0000 0000
                        0201 02ff feb5 8fce ff02 0000 0000 0000
                        0000 0000 0001 0002 8000 0223 002c eb65
                        0100 0002 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000
```

5.5.4.5.1 Test summary

The test was performed with client taken from the KAME LINPORT package. Client sends a solicit message. Server drops the message with “nur DUID received! returning” output. Client timeouts because of the lack of server response.

5.6. DHCPv6 implementation of hp-ux 11i

Source available at:

https://payment.ecommerce.hp.com/cgi-bin/swdepot_parser.cgi/cgi/try.pl?productNumber=DHCPv6&date=

Implemented draft:

unknown (no information in documentation)

5.6.1. Description

The DHCPv6 on HP-UX 11i is a commercial DHCPv6 protocol implementation. It is also available as a trial/free product on HP web pages.

The DHCPv6 product suite enables DHCP servers to pass the configuration parameters using extensions to IPv6 nodes. It automatically allocates reusable network addresses, thereby reducing the cost of managing IPv6 nodes in networks where administrators require more control over the allocation of IP addresses. DHCPv6 is available on HP-UX 11i platform as a web upgrade.

5.6.2. Documentation

The following man pages are distributed with this release of DHCPv6:

- dhcpv6d.1m
  This man page describes dhcpv6d, the DHCPv6 server daemon.
- dhcpv6db2conf.1
  This man page describes dhcpv6db2conf, a utility that is used to convert the client database, dhcpv6client.data into a set of standard configuration variables.
- dhcpv6client_ui.1
  This man page describes dhcpv6client_ui, an interface through which a user contacts the client daemon to obtain IP addresses and other configuration parameters from the server.
- dhcpv6clientd.1m
  This man page describes dhcpv6clientd, the client daemon that is used to obtain configuration parameters from the DHCPv6 servers to configure a host.

Documentation is also available on HP web pages:

- http://www.software.hp.com/cgi-bin/swdepot_parser.cgi/cgi/displayProductInfo.pl?productNumber=DHCPv6
- http://www.docs.hp.com/hpux/pdf/5969-4327.pdf

5.6.3. Functionality

According to the documentation this software supports the following features:

- IPv6 address allocation
- Multiple IP addresses for an interface
- Reconfiguration messages
- Relay mechanism
- Request for configuration parameters from different servers within the same domain
- DNS server address
- DNS suffix
- NTP server address
- NIS domain name
- NIS server address
- NIS+ client domain name
- NIS+ server address
- SLP DA address and its scope
- SLP service scope
- Timezone

5.6.4. Tests

5.7. Motorola Labs DHCPv6

source:

http://www.nal.motlabs.com/livsix/other/ext-server.tar.gz

http://www.nal.motlabs.com/livsix/other/client.tar.gz

date:

May 2001

implemented draft:

draft-ietf-dhc-dhcpv6-18.txt

5.7.1. Description

This software seems to be written by Motorola Labs. Nothing is known about the status or the future of the implementation. The implementation appears to be based on draft 18, but seems to be a limited implementation. It currently works only on Linux.

5.7.2. Documentation

The only documentation is two sample configuration files.

5.7.3. Functionality

It seems to do address delegation only. Addresses can be dynamically allocated from pools, or statically to known hosts based on their link-local addresses.

5.7.4. Tests

5.7.4.1 Environment

Tests were done on Linux with four machines on the same ethernet LAN. One ran the server software, and the other three ran the client software.

5.7.4.2 Compilation

This was straight forward. The client and server tar-files can be extracted and compiled independently. Simply doing "make" in each of their top directories should suffice.

5.7.4.3 Starting the server

First the server needs to be configured. This is relatively simple. The server needs to have a non-EUI-64 address that we specify in the configuration file. Next we define those clients should have static addresses, if we only want to allocate from a pool, this can be skipped. Finally for each link we define prefix, lifetime etc. and also list clients and pools for the links. This might look as follows:

```
server 2001:700:1:0::1:0/64;

client sverresborg
{
     linklocal = fe80::290:27ff:fe50:6bfa;
}

subnet subnet0
{
     prefix = 2001:700:1:0::0/64;
     preferred_lifetime = 0xFFFFFFFF;
     valid_lifetime = 0xFFFFFFFF;
     sverresborg -> 0::42;
     other -> [0::100-0::200];
}
```

this is what was used for the tests. If we store this in a file named test.conf, we can now start the server doing

```
./dhcpv6-server < test.conf
```

There is a lot of output when the server is started, and it outputs a lot of things quite often. It looks something like this:

```
server
client sverresborg
subnet subnet0
thread: 0
pump ex
test
thread: 1
pump ex
thread: 2
pump ex
thread: 3
pump ex
thread: 4
pump ex
recv 512
pump forw recv 1 transaction_id: 12336
```

```
pump ok
msg: 48 id: 1
Unknown message
pump ex
recv 512
pump forw recv 1 transaction_id: 12336
pump ok
msg: 48 id: 1
Unknown message
pump ex
```

Instead of listening to the standard DHCP server port, it uses port 6001! The server listens for all traffic to port 6001. Port 6001 might be used by the X-windows System.

5.7.4.4 Starting a client

The client will read the file dhcpv6.conf in the current directory when started. We simply used the file that was included. We tried first to start it on the client defined as "sverresborg" in the server configuration file.

We got the following output from the server:

```
recv 58
pump forw recv 0 transaction_id: 11165
pump ok
msg: 1 id: 0
solicit
transaction id: 11165
client_addr: fe 80 0 0 0 0 0 2 90 27 ff fe 50 6b fa
option code 1 16
Subnet tested subnet0
Client tested other
Client tested sverresborg
client detected: sverresborg
subnet allocated: subnet0
addresse allocated: 20 1 7 0 0 1 0 0 0 0 0 0 0 0 42
size: 84
02 00 2b 9d fe 80 00 00 00 00 00 00 02 90 27 ff fe 50 6b fa 20 01 07 00 \
00 01 00 00 00 00 00 00 00 01 00 00 00 01 00 30 00 00 00 00 00 00 00 01 \
ff ff ff ff ff ff ff ff 00 01 00 40 20 01 07 00 00 01 00 00 00 00 00 00 \
00 00 00 42 ff ff ff ff ff ff ff ff
sendto 84 Interrupted system call
pump ex
recv 58
pump forw recv 4 transaction_id: 62649
pump ok
msg: 3 id: 4
solicit
transaction id: 62649
client_addr: fe 80 0 0 0 0 0 2 90 27 ff fe 50 6b fa
option code 1 16
Subnet tested subnet0
Client tested other
```

```
Client tested sverresborg
client detected: sverresborg
subnet allocated: subnet0
allocate s
new
addresse allocated: 20 1 7 0 0 1 0 0 0 0 0 0 0 0 0 42
size: 84
07 00 f4 b9 fe 80 00 00 00 00 00 00 02 90 27 ff fe 50 6b fa 20 01 07 00 \
00 01 00 00 00 00 00 00 00 01 00 00 00 01 00 30 00 00 00 00 00 00 00 01 \
ff ff ff ff ff ff ff ff 00 01 00 40 20 01 07 00 00 01 00 00 00 00 00 00 \
00 00 00 42 ff ff ff ff ff ff ff ff
sendto 84 Interrupted system call
```

At the client we get the following:

```
link local address: fe:80:00:00:00:00:00:02:90:27:ff:fe:50:6b:fa:
mfa off
pmfl off
srs off
Adding interface: eth0
Option IA
01 00 2b 9d fe 80 00 00 00 00 00 00 02 90 27 ff fe 50 6b fa 00 00 00 00 \
00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 16 00 00 00 00 00 00 00 01 \
00 00 00 00 00 00 00 00 00 00
sendto 58: Interrupted system call
pump
recv 84
debug: pump_recv
debug: mu_nb = 1
check
pump
debug: mu_nb = 0
dhcp advertise: size 84 server_addr: 20 01 07 00 00 01 00 00 00 00 00 00 \
00 01 00 00
Option IA
size 22
03 00 f4 b9 fe 80 00 00 00 00 00 00 02 90 27 ff fe 50 6b fa 20 01 07 00 \
00 01 00 00 00 00 00 00 00 01 00 00 00 01 00 16 00 00 00 00 00 00 00 01 \
00 00 00 00 00 00 00 00 00 00
sendto 58: Resource temporarily unavailable
pump
recv 84
debug: pump_recv
debug: mu_nb = 1
check
pump
debug: mu_nb = 0
dhcp reply: size 84 server_addr: 20 01 07 00 00 01 00 00 00 00 00 00 00 \
01 00 00
IA DUID: 00 00 00 00 00 00 00 01
Allocated Address: 1
address status:0 preferred lifetime:ffffffff valid lifetime:ffffffff
20 01 07 00 00 01 00 00 00 00 00 00 00 00 42 /64
adding address
```

The client will stay running. Doing "ifconfig" we can see that the address was correctly added.

If we stop the client, we get following output:

```
SIGINT
Option IA
delete address
removing address
size 48
08 00 41 f3 fe 80 00 00 00 00 00 00 02 90 27 ff fe 50 6b fa 20 01 07 00 \
00 01 00 00 00 00 00 00 00 01 00 00 00 01 00 30 00 00 00 00 00 00 00 01 \
ff ff ff ff ff ff ff ff 00 01 00 40 20 01 07 00 00 01 00 00 00 00 00 00 \
00 00 00 42 ff ff ff ff ff ff ff ff
sendto 84: Success
Killed
```

and we can see with ifconfig that it was correctly removed. The server will give the following output:

```
recv 84
pump forw recv 3 transaction_id: 16883
pump ok
msg: 8 id: 3
release
transaction id: 16883
client_addr: fe 80 0 0 0 0 0 2 90 27 ff fe 50 6b fa
option code 1 30
desallocate in subnet0
merging
size: 58
07 00 41 f3 fe 80 00 00 00 00 00 00 02 90 27 ff fe 50 6b fa 20 01 07 00 \
00 01 00 00 00 00 00 00 00 01 00 00 00 01 00 16 00 00 00 00 00 00 00 01 \
ff ff ff ff ff ff ff ff 10 00
sendto 58 Interrupted system call
```

Using tcpdump we see the following when starting the client:

```
   14:36:29.210695 fe80::290:27ff:fe50:6bfa.33941 > ff02::1:2.6001:  udp 58
[hlim 1]
   14:36:29.211439 fe80::290:27ff:fe22:7186.33246 > fe80::290:27ff:fe50:6bfa
.6000:  udp 84
   14:36:29.721750 fe80::290:27ff:fe50:6bfa.33941 > ff02::1:2.6001:  udp 58
[hlim 1]
   14:36:29.722456 fe80::290:27ff:fe22:7186.33246 > fe80::290:27ff:fe50:6bfa
.6000:  udp 84
```

and

```
   14:37:11.088595 fe80::290:27ff:fe50:6bfa.33941 > ff02::1:2.6001:  udp 84
[hlim 1]
   14:37:11.089634 fe80::290:27ff:fe22:7186.33246 > fe80::290:27ff:fe50:6bfa
.6000:  udp 58
```

When stopping it. This seems okay except that port 6000 and 6001 are used instead of 546 and 547 in the draft. This can easily be changed though.

5.7.4.5 More tests

Next we started another client. This was not a known client, and we successfully got address 2001:700:1::100 which is the first address in our defined pool. Starting yet another client, that was allocated 101. Then, we stopped that client and 101 was removed. We also restarted the server. Next we tried to start the client again, and it was then allocated 100, which was still in use by the client that got 100 in the first place. This shows that no state is preserved when the server is restarted, and it doesn't check if an address is in use. So we would say that this server is only usable for known clients.

5.7.4.6 Conclusion

This implementation is highly experimental. It's okay for some testing purposes but not really fit for production. It appears to only do address delegations, and it's mostly suited for handing out statically configured addresses to known clients.

6. L2D2: LDAP to DNS and DHCP – a utility to ease IPv6 DNS and DHCP management

As more and more sites are in the process of evaluating IPv6 on a broader level than a tiny testbed – or even deploying it in production –, the manual management of the IPv6 DNS becomes practically impossible. There must be at least some kind of scripts to be developed and used to generate DNS reverse files, instead of the highly error-prone manual creation.

However, instead of a simple tool, one can try to develop a comprehensive system, which can handle the configuration of the Domain Name System as a whole. The internal logic of such a system can prevent the administrators to create erroneous entries that lead to a malfunctioning DNS and at the same time a well-designed tool can hide the details, keeping the management easy.

In designing such a system, one has to keep in mind the requirements of DHCP as well, as the integration of the management of the two systems is natural: DNS associates hostnames to IP addresses (and providing other domain/host related information) while DHCP is responsible for providing IP addresses to client machines, with possibly other network-related configuration data.

6.1. Requirements

In identifying the main elements of such a system, the following simple goals was formulated:

- The data should be stored in a database
- The user interface must be easy to use and platform-independent
- The subsystems should follow the distributed model with a simple signalling mechanism

- The whole system must be as robust as possible – the DNS (and DHCP) are essential services of networks
- In overall, follow the KISS1 "principle" and reuse already existing elements

In the next sections we discuss the details and consequences of these goals, together with the selected methods and solutions.

6.2. The data storage

By storing the data in a database, we can solve several critical data-related issues by simply requesting the given services from the database engine:

- locking issues in case of concurrent writing of the database
- data replication and redundant servers

Also, by relying on a database, we can use the already existing interfaces at reading/writing instead of developing new ones.

The database could be an SQL-based one, but we chose LDAP instead. With an LDAP tree, we can follow the natural structure of DNS and by developing a suitable schema, we can much the LDAP tree to DNS and DHCP as close as possible. Using LDAP, there is no need for special encoding of multivalue properties. There are already exist DNS and DHCP schema from which we could learn much at designing our specialized schema.

6.3. The interface

The user (i.e. DNS/DHCP administrator) interface is very critical for the success of such a tool. If the interface is cumbersome, non-intuitive or not flexible enough, then the system is developed in vain and our efforts are wasted.

At the same time, the interface must be platform-independent because there will always be administrators who prefer Unix as their desktop machine while others run Windows instead.

One of the simplest platform-independent interface is a web based one. Using a CGI interface written in Perl, fast prototyping and rapid developing can be achieved. The experiences of the countless web based interfaces can be used and a fancy system can be designed fairly easily.

6.4. The distributed model

In order to be flexible, we must follow the distributed model; all the components of the system could possibly be installed and run on separate system:

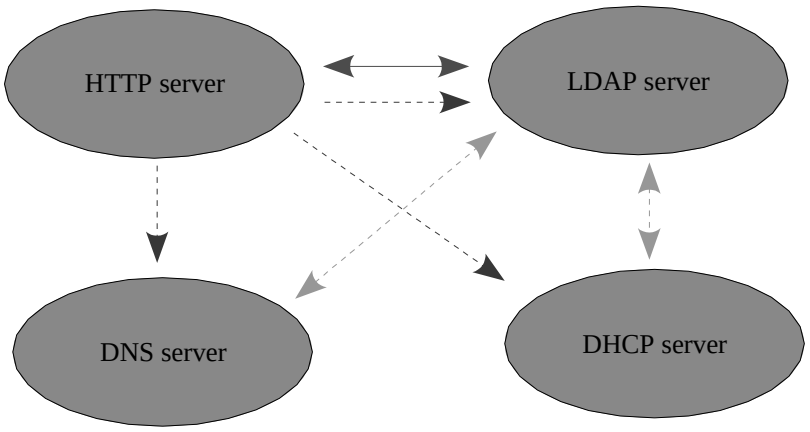

Figure 1 L2D2 architecture

On Figure 1 the red continuous double-ended arrow denotes the single read-write access of the LDAP database from the CGI interface running on the HTTP server. The blue dashed single-ended arrows denote the signals sent from the CGI interface to the services, when they have to refresh their data from the LDAP database. The green dashed double-ended arrows denote the read-only access of the LDAP database when the services refresh their data.

6.5. Robustness

In order to achieve maximal robustness, we do not feed the DNS and DHCP services directly from LDAP. Instead, we run agents on the DNS and DHCP servers, which create the configuration files and signal, restart/reload the daemons when required. Thus we can guarantee, that the DNS and DHCP services can run uninterrupted when the LDAP server is down or the additional LDAP lookup does not slow down the services

On the machines running the actual DNS, DHCP and LDAP servers the agents are not needed to be run continuously: they can be started from `inetd`. Because we have to send a signal only to trigger the refreshing of the data from the LDAP database, we can even use UDP as transport protocol with a simple message content something like:

<zone name> [DNS] [DHCP] [LDAP]

which means for the agent: the subtree of the given zone changed in the LDAP database. (Optionally the message can instruct which service needs updated only.) The agent then creates the configuration files for the service and then restart/signal the daemon. The agent creates

- LDAP access control entries when new zones and administrators are introduced

- DNS zone and reverse zone files
- DHCP configuration file

6.6. Keep it simple...

Focusing on the KISS principle, we build on what is the closets to "Software ICs[1]": Perl and Perl modules from CPAN (Comprehensive Perl Archive Network). Thus we do not have to reinvent the usage of an LDAP database; the CGI interface routines and how to create HTML expressions; IPv4/IPv6 address manipulation functions, and so on. The CGI interface and the agents are all written in Perl.

6.7. Security considerations

All the communications channels except the signalling are protected with SSL/TLS. In theory some kind of DoS attack could be attempted by misusing the signalling mechanism, however the agents create new configuration files only when the LDAP tree has really changed since the last generation of the configuration files.

6.8. Supported functionalities

The following functionalities of the Domain Name System are supported:

- Zones can be defined together with the corresponding IPv4 and IPv6 address spaces.
- Zone administrators can delegate sub-zones to other administrators, which zones then stored in subtrees of the LDAP database. Stealth nameservers in the sub-zones can be introduced (even the primary nameserver might be a stealth one.)
- At the beginning we support data corresponding to the DNS resource records SOA, NS, MX, A, AAAA, CNAME, TXT, HINFO and SRV. Reverse tables and PTR records are generated automa(t|g)ically.
- Classless delegation of IPv4 subnets smaller than /24 is supported.
- Subnets (subdomains) without delegation can be introduced as well.

According to RFC3364, we do not attempt to generate A6 or DNAME resource records and we use the hexadecimal representation of IPv6 addresses instead of bit labels.

From DHCP point of view, the following functionalities are supported:

DHCP subnet and host entries as key elements

- Shared networks
- Groups of hosts, subnets and shared networks
- Most important DHCP parameters are directly supported: range declarations, lease time specifications, hardware address, default routes, etc.
- Arbitrary DHCP parameter statements are supported via a generic option

According to the initial design, groups of groups cannot be specified.

[1] The term was the title of Brad Cox and Lamar Ledbetter's classical article in *Byte.*

6.9. Current status

The structure and the elements of the whole system are identified and fixed. In supporting the requirements of the project, we completed the IPv6 support in the excellent `NetAddr::IP` Perl modul – the author accepted our patches and the new version with IPv6 support has already be released on CPAN. The LDAP schema, after rewriting several times, is due to be finished. The frameworks of the agents are ready: we are working on the modules writing OpenLDAP ACLs and zone files according to bind9. The work on the CGI interface is also in progress. The current release plan for the whole system is at the end of the first quarter of 2003.

6.10. Availability

The complete L2D2 system will be available from the URL `http://www.kfki.hu/cnc/project/l2d2/` under the GPL licence.

7. Conclusion

In this report we summarised the standardisation status DHCPv6, and we found promising. We described the possible usage of DHCPv6 complementing autoconfiguration or without autoconfiguration. We reported our tests about the status and usability of the available DHCPv6 implementation. The result is rather disappointing. Since we felt that operation of DHCPv6 in concert with DNS can be difficult, therefore L2D2 is under development in the 6NET project to ease the administrative burden of IPv6 administrators.

References

[DHCP.ORG] Resources for DHCP, web page: http://www.dhcp.org/

[DHCP-HANDBOOK] Ralph Droms, Ted Lemon, " DHCP Handbook", Macmillan Technical Publishing, ISBN: 1-57870-137-6 , see more http://www.dhcp-handbook.com/

[IETF-DHC] IETD Dynamic Host Configuration Charter, http://www.ietf.org/html.charters/dhc-charter.html