Project Number:	**IST-2001-32603**
Project Title:	**6NET**
CEC Deliverable Number:	**32603/WWU(JOIN)/DS/3.2.3v2/A1**
Contractual Date of Delivery to the CEC:	15 November 2004
Actual Date of Delivery to the CEC:	17 January 2003
Title of Deliverable:	DHCPv6 implementation and test report version 2
Work package contributing to Deliverable:	WP3
Type of Deliverable*:	RP
Deliverable Security Class**:	PU
Editors:	Christian Schild, André Stolze
Contributors:	Bartosz Belter, Michał Balcerkiewicz, Ralph Droms, Francis Dupont, Bartosz Gajda, József Kadlecsik, Marcin Kamiński, János Mohácsi, Blazej Pietrzak, Stig Venaas, Christian Schild, André Stolze
Reviewers:	Janos Mohacsi, Bartek Gajda, João Nuno Ferreira

* 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 developments in DHCP6 implementations since the last version of this document. There are several RFC concerning different types of DHCPv6 servers. Only some of them could be considered complete and working for today. As DHCPv6 is a vital part for easy administration these tools urgently need further development.

Keywords: stateful and stateless DHCP, IPv6, autoconfiguration, testing, DNS, network management, prefix delegation, DHCPv6 relay agent

Table of Contents

1 Introduction

After many years of work, DHCPv6 was published as RFC3315 [RFC3315]. Despite the existence of stateless autoconfiguration for IPv6 (RFC 2462 [RFC2462]), there is still a need for DHCP. On the one hand, it complements stateless autoconfiguration where it can supply hosts with DNS, NTP and other configuration data. On the other hand, a network administrator might want to gain more control over the IP addresses used than is possible with stateless address configuration. A stateful DHCPv6 implementation as of RFC 3315 offers both. In addition, the IETF DHC working group [DHC] published a more lightweight respective "stateless" DHCPv6 version (RFC 3736 [RFC3736]), which serves only as a source for configuration options that are not already delivered to the host with stateless autoconfiguration.

1.1 Using DHCPv6 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, anycasting and additional autoconfiguration options). DHCP was 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. RFC 3736 offers a solution for this. There are other features more 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 DHCPv6 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 fulfil 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 were several 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 do not require extensive revision to become compliant with the final version of the specification.

One question 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 3.3, is used to provide these additional configuration parameters.

DHCP for IPv6 was published as an Internet Proposed Standard in June 2003 in RFC 3315 "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" [RFC3315]. It describes the complete "stateful" DHCPv6 implementation. Derived from this RFC and from several subsequent RFCs describing DHCPv6 options an additional light-weight specification of a "stateless" DHCPv6 version was published as an Internet Proposed Standard in April 2004 in RFC 3736 "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6" [RFC3736].

3 Overview of the DHCPv6 standards

3.1 Basic "stateful" operation (RFC 3315)

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 RFC 3118 [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.

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 3.3, stateless DHCPv6 service requires only a subset of the mechanism and messages of the full DHCPv6 protocol, and is easier to implement and deploy.

3.2 Additional DHCPv6 Options

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

Transferring information as separate options to the clients gives the opportunity to add more options for additional information at a later point of time. Further options are already published as proposed standard RFCs, while others are still in development with different levels of maturity. They are described in short in the remainder of this section.

3.2.1 SIP server configuration option for DHCPv6 (RFC3319)

RFC 3319 ("Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers") [RFC3319] defines a DHCPv6 option that contains a list of domain names or IPv6 addresses that can be mapped to one or more Session Initiation Protocol (SIP) outbound proxy servers. It was published as a Proposed Standard in July 2003.

3.2.2 IPv6 prefix options for DHCPv6 (RFC 3633)

The "Prefix Option" is used for prefix delegation in DHCPv6. An ISP uses prefix delegation to delegate a prefix or prefixes to a customer. 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.

This option was published as Proposed Standard RFC 3633 "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6" [RFC3633] in December 2003.

3.2.3 DNS configuration options for DHCPv6 (RFC 3646)

"DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" was published as Proposed Standard RFC 3646 [RFC3646] in December 2003. It defines two DNS configuration options. 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.

3.2.4 NIS/NIS+ configuration options for DHCPv6 (RFC3898)

A third kind of DHCPv6 option that is already published as a Proposed Standard is described in RFC 3898 "Network Information Service (NIS) Configuration Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" [RFC3898] published in October 2004. It defines four different options that convey a list of NIS/NIS+ servers and NIS/NIS+ domain names to a client.

3.2.5 SNTP server option for DHCPv6

The draft "Simple Network Time Protocol Configuration Option for DHCPv6" (<draft-ietf-dhc-dhcpv6-opt-sntp-00> [SNTP]) describes a new DHCPv6 option for passing a list of SNTP server addresses to a client. It is currently still under IESG review and will hopefully be published as a Proposed Standard RFC.

3.2.6 Lifetime option for DHCPv6

The draft "Information Refresh Time Option for DHCPv6" defines an upper bound for how long a client should wait before refreshing information retrieved from DHCPv6. It is under discussion in IETF's DHC working group and is currently published with draft status as <draft-ietf-dhc-lifetime> [LIFETIME].

3.2.7 Client FQDN option for DHCPv6

Draft <draft-ietf-dhc-dhcpv6-fqdn> [CLIENTFQDN] specifies a "DHCPv6 Client FQDN Option" and is under discussion in the DHC working group. This option can be used to exchange information about a DHCPv6 client's fully qualified domain name and about responsibility for updating DNS resource records (RRs) related to the client's address assignments.

3.2.8 Other DHCPv6 options

There were several other DHCPv6 options discussed in the past in the DHC WG. Several of them were expired in the last year, but might get picked up again at a later point of time. A short list of what was available:

- DSTM Options for DHCPv6 (<draft-ietf-dhc-dhcpv6-opt-dstm-02>) [DSTM]
- DSTM Ports Option for DHCPv6 (<draft-ietf-dhc-dhcpv6-opt-dstm-ports>) [DSTMPORT]

- Client Preferred Prefix option for DHCPv6 (<draft-ietf-dhc-dhcpv6-opt-cliprefprefix>) [CLIPREF]
- Load Balancing for DHCPv6 (<draft-ietf-dhc-dhcpv6-loadb>)[LOADB]

3.3 Stateless operation (RFC 3736)

The DHCPv6 service of providing configuration information without address assignment is called "stateless DHCPv6" ([RC3736]), 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 [RFC3315] 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 PE router, using, for example, DNS configuration information configured by the PE 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.

Nodes which have obtained IPv6 addresses through some other mechanism, such as stateless address autoconfiguration [RFC2462] or manual configuration can use stateless DHCP to obtain other configuration information such as a list of DNS recursive name servers or SIP servers. A stateless DHCP server provides only configuration information to nodes and does not perform any address assignment.

Clients and servers implement *Information-Request* and *Reply* messages for stateless DHCP service. *Information-Request* message is sent by a DHCP client to a server to request configuration parameters. *Reply* message is sent by DHCP server to the client and contains configuration parameters. Additionally, servers and relay agents implement *Relay-forward* and *Relay-reply* messages. *Relay-forward* is sent by a DHCP relay agent to carry the client message to a server. *Relay-reply* message is sent by a DHCP server to carry a response message to the relay agent [RFC3736].

The basic RFC [RFC3315] and subsequent RFCs define various options. For a stateless operation it is required to implement a specific set of them:

Clients and servers implement the options shown in table 1 for stateless DHCP service.

Table 1. Options implemented by stateless clients and servers

Option Request	specifies the configuration information that the client is requesting from the server
Status Code	used to indicate completion status or other status information.
Server Identifier	used to identify the server responding to a client request.

Servers and relay agents implement the options shown in table 2 for stateless DHCP service.

Table 2. Options implemented by servers and relay agents

Client message	sent by a DHCP relay agent in a Relay-forward message to carry the client message to a server
Server message	sent by a DHCP server in a Relay-reply message to carry a response message to the relay agent
Interface-ID	sent by the DHCP relay agent and returned by the server to identify the interface to be used when forwarding a message to the client

Clients and servers implement the options shown in table 3 to pass configuration information to clients.

Table 3. Options implemented by clients and servers to pass configuration information to clients.

DNS Recursive Name Servers	specifies the DNS recursive name servers the client uses for name resolution
DNS search list	specifies the domain names to be searched during name resolution
SIP Servers	specifies the SIP servers the client uses to obtain a list of domain names of IPv6 addresses that can be mapped to one or more SIP outbound proxy servers

These additional options are not part of the basic RFC [RFC3315], but are published as separate RFCs. When more additional options are published, they might get added to this list.

Clients and servers may implement the options shown in table 4 for stateless DHCP service.

Table 4. Options that may be implemented by clients and servers.

Preference	sent by a DHCP server to indicate the preference level for the server.
Elapsed time	sent by a DHCP client to indicate the time since the client began the DHCP configuration process.
User Class	sent by a DHCP client to give additional information to the server for selecting configuration parameters for the client
Vendor Class	sent by a DHCP client to give additional information about the client vendor and hardware to the server for selecting configuration parameters for the client.
Vendor-specific Information	used to pass information to clients in options defined by vendors.
Client Identifier	sent by a DHCP client to identify itself; clients are not required to send this option; servers send the option back if included in a message from a client.
Authentication	used to provide authentication of DHCP messages.

DHCP servers that are intended only for stateless configuration may receive messages from clients that are performing stateful address configuration. A DHCP server that is only able to provide stateless configuration information through an *Information-request/Reply* message exchange discards any messages other than *Information-Request* or *Relay-forward* it receives, and the server

does not participate in any stateful address configuration message exchanges. If there are other DHCP servers that are configured to provide stateful address assignment, one of those servers will provide the address assignment [RFC3736].

3.4 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 relay agents to forward requests between client and server
- 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

There are two different types of DHCPv6 implementations, stateful and stateless, based either on RFC 3315 or RFC 3736. While stateful DHCPv6 implementations are supposed to include the whole DHCPv6 feature set as of RFC 3315 (including address delegation to the client), stateless DHCP implementations only convey additional configuration parameters to the client. As the name suggests, stateless DHCPv6 servers do not need to keep a state to client. They are easier to implement and can even be placed on a providers edge router.

In addition to the list below, there are several other implementations that are nowadays outdated. They are based on older draft versions of DHCPv6 and as there were significant changes in late versions of that draft they are not compliant to the final RFC. They were tested in the beginning of the 6NET project early 2003. Test results can be found in the first version of this deliverable (D3.2.3v1) [D3.2.3v1].

4.1 Stateful DHCPv6 implementations

4.1.1 NEC Netlab

NEC's product implements a DHCPv6 client, a stateful server and a relay. While the client and server are not freely available, the DHCPv6 relay is published under the GNU public licence. It implements all options as of RFC 3315 and offers the means to statefully assign addresses to the clients. Additionally it implements the following options:

- IPv6 prefix option
- DNS configuration option

- SIP server configuration option
- Lifetime option
- Time configuration option

NEC offered an exclusive cooperation to some 6NET partners to test and enhance their implementation. This cooperation is under NDA terms and not all of the results can be published here.

4.1.2 Sourceforge.net

This implementation is an open source project under the BSD license available for Linux hosts. It contains both a client and server but no relaying functionality yet.

4.1.3 Dibbler

Dibbler is a portable DHCPv6 implementation. Currently Linux 2.4/2.6 and WindowsXP ports are being actively developed. In the not so distant future, BSD version will follow.

Dibbler offers both, a DHCPv6 server and a client [Dibbler]. It was released under GPL licence. The core of Dibbler was written in C++ programming language and as the author claims is fully portable. The rest of code is system dependent and was written in plain C. Currently Dibbler compiles with gcc tool and also using Visual Studio environment.

4.1.4 HP-UX

HP has implemented an advanced DHCPv6 client, server and relay. It is only available for HP-UX and is available for their latest HP-UX (11iv1) operating system. It implements the full set of functionality as of RFC 3315 and uses the following additional options:

- SIP server configuration option
- DNS configuration option
- NIS/NIS+ configuration option

4.2 Stateless DHCPv6 implementations

4.2.1 Kame

Kame-dhcp6 is an open-source implementation of Dynamic Host Configuration Protocol for IPv6 (DHCPv6) developed by the KAME project [KAME-DHCPv6].

The KAME implementation claims to conform to RFC 3315 but does implement address delegation, which makes it more a stateless DHCPv6 like specified in RFC 3736. The KAME developers give the following statement in this issue:

> ‘Note that the current implementation does not support IPv6 address allocation by DHCPv6, and there is no plan to implement that feature at the moment. The main purpose of this implementation is to provide a way of IPv6 prefix delegation (RFC3633) and to provide some "stateless" configuration information such as DNS recursive server addresses.‘

The KAME implementation offers client, server and relay. It uses the following additional options:

- SIP server configuration option
- IPv6 prefix option
- DNS configuration option

4.2.2 Cisco

The Cisco DHCPv6 implementation, which was introduced in Cisco IOS Software Release 12.3(4)T, runs on Cisco routers. Both client and server are specifically intended to provide the prefix delegation feature and therefore do not implement the entire DHCPv6 protocol. At present, Cisco's DHCPv6 implements prefix delegation, the rapid-commit mechanism and stateless DHCPv6 and all basic options as of RFC 3315.

4.2.3 Juniper

Juniper introduced DHCPv6 functionality in their router operating system JunOS version 5.3. Only the DHCPv6 server is available. It uses the prefix delegation option and the DNS configuration option.

4.2.4 Hitachi

Hitachi has implemented a DHCPv6 server for its GR2000 platform. It conveys the following options to a client:

- IPv6 prefix delegation
- DNS configuration option
- Time configuration option

5 Test report of the DHCPv6 implementations

5.1 NEC netlab

5.1.1 Description

The DHCPv6 implementation of the NEC netlabs is a pre-production release given to the JOIN team for testing purposes under a NDA. So the description in this document will be kept more or less abstract.

5.1.2 Documentation

The testing version JOIN got included a .pdf file of documentation for each of the three programs. For the relay there is also a man page (**`dhcpv6c(8)`**) installable.

5.1.3 Functionality

In the server and client the following RFCs/drafts are fully implemented:

- Dynamic Host Configuration Protocol for IPv6 (DHCPv6) (RFC 3315)
- IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6 (RFC3633)
- DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6) (RFC3646)
- Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers (SIP)(RFC3319)
- draft-ietf-dhc-lifetime-00.txt
- draft-ietf-dhc-dhcpv6-opt-timeconfig-03.txt
- draft-ietf-dhc-dhcpv6-stateless-04.txt

For the relay agent there was only RFC 3315 to keep in mind.

5.1.4 Tests

The software supports both prefix delegation and address assignment, but tests were only done for address assignment.

5.1.4.1 Environment

All tests were executed on Debian Linux (testing) with 2.8.1 or 2.4.24 kernels.

5.1.4.2 Compiling the code / Installing the software

The package comes without source code, so there was nothing to compile. All there is to be done is to extract the .tgz and do a ./configure in one of the directories SERVER, CLIENT or RELAY-AGENT.

5.1.4.3 Test 1

In the first test the aim was to configure hosts on the same link as the server was on. In this case there was noting to do except of adjusting the address range in the server's configuration file. In this scenario clients and server worked as it was expected.

5.1.4.4 Test 2

In the second test the scenario was changed in a way that the server was on another link than the clients. After some smaller configuration problems this scenario worked well.

5.1.4.5 Conclusions

Although the software is described as "not yet ready to sell" by NEC it was very nice to configure and run. Nevertheless the software worked, as it had been expected.

5.2 Sourceforge.net

5.2.1 Description (from the software itself)

This implementation supports IPv6 address assignment to the clients. It also has the support for prefix delegation, DNS server updates but those features are not validated yet.

The Linux implementation is based upon KAME's DHCPv6 implementation on BSD, which lacked the support for dynamic address assignment feature, now supported in Linux.

5.2.2 Documentation

The authors as man pages provide documentation.

5.2.3 Functionality

The authors of the software provide the following feature list:

A. Validated Features

- IPv6 address Assignment & Prefix Delegation
- Server configuration file support for both static and dynamic assignments.
- Server lease file support for saving the entire client's IPv6 address binding info.
- Client IPv6 address assignment and temporary IPv6 address assignment support on the same link.
- Supported Options: Rapid commit, Server Preference, Information Request, Unicast,
- Elapsed Time, ClientID, ServerID, IA_NA, IA_TA, IA_ADDR, IA_PD, Status support.
- Solicit/Request/Advertise/Reply/Information-request messages/Renew/Rebind/Release/Confirm/ messages support for IPv6 address binding.
- Client configuration file support for IPv6 addresses assignment.
- Client lease file support for saving individual client ipv6address binding info.
- DNS servers update support according to draft-ietf-dhc-dhcpv6-opt-dnsconfig-03.txt.
- Prefix delegation support according to draft-ietf-dhc-dhcpv6-opt-prefix-delegation-03.txt
- radvd.conf update and radvd reload for prefix delegation

B. Support available but not validated yet

- Request option support

C. ToDo List

- Authentication/User class/Vendor class/Interface-ID option support
- Relay agent support.
- Reconfig/Relay messages support.

5.2.4 Tests

5.2.4.1 Environment

As in most of the tests the platform was Debian Linux (testing) with 2.8.1 and 2.4.24 kernels.

5.2.4.2 Compiling the code/ Installing the software

This is how it is described and it woks fine.

1. tar -xvzf dhcp6.tar
2. ./configure
3. make
4. insmod ipv6 (if ipv6 is not compiled in the kernel)
5. mkdir /var/db/dhcpv6 (create dhcpv6 database directory)
6. cp dhcp6c.conf /etc/dhcp6c.conf (copy client configuration file)
7. cp dhcp6s.conf /etc/dhcp6s.conf (copy server configuration file)
8. ./dhcp6s -dDf [eth0 eth1 ...] (start server, turn on debug)
9. ./dhcp6c -dDf eth0 (start client, turn on debug)

5.2.4.3 Tests

Because there is no relaying support the only test that could be done was to test DHCPv6 on the same link. Client and server worked fine when using configuration built up by information provided by the man pages.

Server configuration file:

```
interface eth0 {
        server-preference 255;
        renew-time 60;
        rebind-time 90;
        prefer-life-time 130;
        valid-life-time 200;
        allow rapid-commit;
        option dns_servers 2001:638:500:101::53 join.uni-muenster.de;
        link AAA {
                pool{
                      range 2001:638:500:101::1111 to 2001:638:500:101::9999/64;
                      prefix 2001:638:500::/48;
                };
};
```

Client config:

```
Interface eth0{}
```

5.2.4.4 Conclusions

Because of the missing but planed relay-agent the software is not yet usable for stateful operation in an efficient way. But we look forward that it will be if the relay is completely implemented. Stateless operation for this implementation was not fully tested.

5.3 Dibbler

5.3.1 Description

Dibbler is a client/server DHCPv6 solution that works for now just on a single link, but it is planned to extend functionality towards relaying support.

5.3.2 Documentation

The binary packages contain two .pdf files containing the complete documentation of the current software. Using this it is very easy to configure and run it.

5.3.3 Functionality

The feature list as it can be found on the web site http://klub.com.pl/dhcpv6/

- MESSAGES: SOLICIT, ADVERTISE, REQUEST, REPLY - many servers support, client can be configured to ask specific addresses. Client can be configured to ask for arbitrary number of addresses. One client can be serviced by multiple servers (e.g. client asks for 5 addresses, preferred server can lease only 3, so client sends request for remaining 2 addresses to backup server).
- MESSAGES: RENEW/REBIND/REPLY - fully configurable addresses/options renewal.
- MESSAGES: DECLINE/REPLY - Duplicate Address Detection (DAD) is fully supported.
- MESSAGES: CONFIRM - supported by server, client-side support in progress
- MESSAGES: INFORMATION-REQUEST/REPLY - stateless autoconfiguration
- OPTIONS:IA - standard addresses assignment. Client can ask for specific addresses, multiple IAs per one message are supported
- OPTIONS:RAPID-COMMIT - expedited configuration (SOLICIT/REPLY)
- OPTIONS:UNICAST - messages can be exchanged using unicast communication instead of multicast.

- OPTIONS:PREFERENCE - you can start multiple servers and configure them to have preference between 0-255.

- OPTIONS:all required options (SERVERID,CLIENTID etc.) are supported

- ADDITIONAL OPTIONS:DNS Servers (works in Linux only, Microsoft didn't released API to support DNS in windows environment)

- ADDITIONAL OPTIONS:SearchDomains (works in Linux only)

- ADDITIONAL OPTIONS:TimeZone (data dumped to file)

- ADDITIONAL OPTIONS:NTP Server (data dumped to file)

- OTHER: multiple servers support.

- OTHER: data is stored in XML, so Dibbler is easily scriptable.

- OTHER: architecture is layered - upper, fully portable is written in C++, lower, system-specific is written in C. Porting to other system/architecture requires implementing only less then 10 low-level functions (e.g. IPv6 address adding).

5.3.4 Test of stateful operation

5.3.4.1 Environment

The software was tested on Debian Linux (i386 testing tree) and Microsoft Windows XP in a way that every OS has been client and server with different clients.

5.3.4.2 Compiling the code

There were working binary packages available for each of the involved Systems so there was no need to compile the software. Just unpack edit the configuration files as described in the README and use.

5.3.4.3 Test

Because there is no relaying support the only test that could be done was to test DHCPv6 on the same link. In this case we used both a Linux and a Windows XP client for each a Linux and Windows XP server. In any case there was the same result: Everything the client and server supports worked fine but there were problems with some Windows XP versions. There occurs just one smaller issue. On every client the address ::/128 has been assigned to the configured interface, which might lead to some problems while using applications using that address. Additionally it is to be said that the domain option or better the server does not work with domains containing a "-" in it.

Server configuration file:

```
log-mode short
iface eth0
{
   option dns-server 2001:638:500:101::53
   option domain join.unimuenster.de
   iface-max-lease 50
   client-max-lease 5
   preference 7
   class
   {
       T1 60-120
       T2 600-1200
       prefered-lifetime 1800
       valid-lifetime 3600
       class-max-lease 20
       pool 2001:638:500:101::aaaa-2001:638:500:101::ffff
   }
}
```

Client configuration file:

```
log-mode short
iface eth0
{
   option dns-server
   option domain
   IA
}
```

5.3.4.4 Conclusions

Because of the missing but planed relay-agent the software is not yet usable for stateful operation in an efficient way. But we look forward that it will be if the relay is completely implemented.

5.3.5 Test of stateless operation

5.3.5.1 Testing methodology

Tests were designed to check the conformance with RFC 3315 [RFC3315] and RFC 3736 [RFC3736] standards of Dibbler [Dibbler] client and server implementation on Windows XP and Linux platforms. To achieve this, the goals shown in table 5 were set.

Table 5. Test goals

Goal	Description
Client message format	• Check if the message structure is RFC3315 [RFC3315] and RFC3736 [RFC3736] conformant.
Transmission of messages	• Check if the client sends messages using multicast. • Check if the client sends only *Information-Request* messages to server. • Check if the client can send *Information-Request* asking for DNS servers, DNS search domains and SIP servers. • Check if the client can send correct *Information-Request* message after receiving *Reply* message from server. • Check if the client implements *Preference* option. • Check if the client implements *User Class* option. • Check if the client implements *Vendor Class* option. • Check if the client implements *Vendor-specific Information* option. • Check if the server can reply for DNS servers, DNS search domains and SIP server's requests from client. • Check if the server responds only with *Reply* messages to client. • Check if the server implements *Client Identifier* option. • Check if the server implements *Preference* option. • Check if the server implements *Vendor-specific Information* option. • Check if server implements communication with relay agents.
Reliability of client initiated message exchanges	• Check the retransmission strategy to be used by clients when no respond is from the server.
Authentication	• Check if the client implements Authentication option. • Check if the server implements Authentication option. • Check authentication of DHCP messages.

Goal	Description
Interaction with operating system	• Check if the client can set DNS servers received from the server in the operating system environment. • Check if the client can set domain search list received from the server in the operating system environment. • Check if the client can set SIP servers list received from the server in the operating system environment. • Check if the client can clean-up the system when client is shut down (killed).

5.3.5.1.1 Compiling the code

While testing, the most recent version of Dibbler [Dibbler] was used (0.3 RC1). Linux version has a serious bug that prevents program from running. In order to complete tests the bug had to be found and fixed. The author was also informed about this inconvenience. TClntTransMgr class (CintTransMgr.cpp file) in method openLoopbackSocket does not set the list pointer to the beginning of the list of network interfaces. That prevents application from getting interfaces correctly and causes it to exit with error. In order to fix it one must add

```
IfaceMgr->firstIface();
```

right after the #ifndef WIN32 clause.

Windows version of Dibbler compiles pretty smooth and no problems were noticed.

5.3.5.1.2 Testing Environment and Tools

Testing environment consisted of four computers running Linux and Windows operating systems in a separated network in PSNC. In case of Linux, Slackware 10.0 distribution was used with kernel version 2.8.1 for x86 platform. The other operating system was Windows XP Professional with Service Pack 2. To avoid any problems the firewalls were turned off. Three computers were DHCPv6 stateless servers and one was DHCPv6 client.

Dibbler was tested with the latest packet capture program called Ethereal [Ethereal]. Ethereal has support for IPv6 and in general has proved to be all RFCs complaint.

All packets and their fields were checked against RFC 3315 – Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC3315].

5.3.5.2 Test results for Dibbler client

This chapter contains test results obtained from testing the Dibbler client for RFC 3736 [RFC3736] conformance.

5.3.5.2.1 Get DNS servers test

The aim of this test is to check if dibbler client can send correct *Information-request* message asking for DNS server address(es) and append DNS servers returned from server's *Reply* message to the system servers list.

- ***Status***

  Test failed.

  On Linux if resolv.conf doesn't end with CR the resulting resolv.conf is corrupted, because Dibbler [Dibbler] doesn't append CR before appending DNS-server list.

- ***Preconditions***

  Client didn't send any message before this test.

- ***Input***

  client.conf:

```
log-mode short
iface eth0
{
    stateless
    option dns-server
    T1 60
    T2 60
}
```

- ***Expected output***

  Client should send *Information-request* message containing DNS recursive name server option. Client should use multicast. Server should respond with *Reply* message containing DNS server's addresses.

Client should then update DNS servers list.

- ***Actual output***

  Message sent to server:

```
Destination address: ff02::1:2 (ff02::1:2)
Message type: Information-request (11)
    Transaction-ID: 0x00364437
    Client Identifier
        option type: 1
        option length: 14
        DUID type: link-layer address plus time (1)
        Hardware type: 0
        Time: 1099494430
        Link-layer address
    Elapsed time
        option type: 8
        option length: 2
        elapsed-time: 1 sec
    Option Request
        option type: 6
        option length: 4
        Requested Option code: DNS recursive name server (23)
        Requested Option code: Unknown (42)
```

Message received from server:

```
Message type: Reply (7)
    Transaction-ID: 0x00364437
    Client Identifier
        option type: 1
        option length: 14
        DUID type: link-layer address plus time (1)
        Hardware type: 0
        Time: 1099494430
        Link-layer address
    DNS recursive name server
        option type: 23
        option length: 32
        DNS servers address: 2000::100
        DNS servers address: 2000::101
    Server Identifier
        option type: 2
        option length: 14
        DUID type: link-layer address plus time (1)
        Hardware type: 0
        Time: 1099923422
        Link-layer address
```

5.3.5.2.2 Get DNS servers when it was already obtained from server test

The aim of this test is to check if client can send correct *Information-request* message asking for DNS server address(es) and append DNS servers returned from server's *Reply* message to the system DNS servers list after at least one *Information-request* message was sent prior test.

- ***Status***

Test failed.

Client uses multicast. On Linux and Windows implementation no message was received from the server. Message sent to the server is RFC 3315 [RFC3315] complaint, but it did not contain any option. The server did not respond to the *Information-request* message, which is correct behaviour, because there was no option in the *Information-request* message.

- ***Preconditions***

Client sent *Information-request* message before this test.

- ***Input***

Configuration file (client.conf):

```
log-mode short
iface eth0
{
    stateless
    option dns-server
    T1 60
    T2 60
}
```

- ***Expected output***

Client should send *Information-request* message containing DNS recursive name server option. Client should use multicast.

Server should respond with *Reply* message containing DNS servers addresses.

Client should then update DNS servers list.

- ***Actual output***

Message sent to server:

```
Destination address: ff02::1:2 (ff02::1:2)
Message type: Information-request (11)
Transaction-ID: 0x00364437
Client Identifier
    option type: 1
    option length: 14
    DUID type: link-layer address plus time (1)
    Hardware type: 0
    Time: 1099494430
    Link-layer address
Elapsed time
    option type: 8
    option length: 2
    elapsed-time: 2 sec
```

5.3.5.2.3 Client shut down test

Dibbler client application is released (killed).

- ***Status***

Test failed.

On Linux client removes from resolv.conf DNS servers and domain search list received from server. Unfortunately it also changes the resolv.conf's access control list to 0014. Although removal of DNS servers and domain search list received from server is desired, access control list should not be changed, so it is considered as a bug.

- ***Preconditions***

Dibbler client changed system configuration (received at least one REPLY message from server). resolv.conf contains DNS server list received from dibbler client.

- ***Input***

client.conf:

```
log-mode short
iface eth0
{
    stateless
    option dns-server
    option domain
    T1 60
    T2 60
}
```

- ***Expected output***

Client restores system state that was prior to running client - removes from resolv.conf (Linux) DNS servers and domain search list received from server. No message is send to server.

Client calls external program netsh.exe with valid parameters and eventually removes DNS entries from the system.

- ***Actual output***

On both platforms client did not send any message to server when being shut down.

- ***Linux***

Client removes from resolv.conf DNS servers and domain search list received from server. Unfortunately it also changes the resolv.conf's access control list to 0014. Although removal of DNS servers and domain search list received from server is desired, access control list should not be changed, so it is considered as a bug.

- ***Windows***

Client calls external program netsh.exe with valid parameters and eventually removes DNS entries from the system.

5.3.5.2.4 Get domain search list test

Client sends *Information-request* to the server asking for domain search list.

- ***Status***

Test failed.

On Linux client sends *Information-request* containing option Domain Search List. The message is complaint with RFC 3315 [RFC3315] and RFC 3736 [RFC3736]. Server responds to the client with *Reply* message containing domain search list. Client updates the domain search list in the resolv.conf, but also adds 0xCD character at the end of the domain search list before the CR character. It is only the case when only one domain search list is returned.

- ***Preconditions***

Client didn't send any message before this test.

- ***Input***

client.conf:

```
log-mode short
iface eth0
{
    stateless
    option domain
    T1 60
    T2 60
}
```

- ***Expected output***

Client sends *Information-request* containing option Domain Search List. Client should use Server responds to the client with *Reply* message containing domain search list. Client updates the domain search list in the system.

- ***Actual output***

Client sent the following message:

```
Destination address: ff02::1:2 (ff02::1:2)
Message type: Information-request (11)
Transaction-ID: 0x001840f8
Client Identifier
    option type: 1
    option length: 14
    DUID type: link-layer address plus time (1)
    Hardware type: 0
    Time: 1099494430
    Link-layer address
Elapsed time
    option type: 8
    option length: 2
    elapsed-time: 1 sec
Option Request
    option type: 6
    option length: 4
    Requested Option code: Domain Search List (24)
    Requested Option code: Unknown (42)
```

Server replied with the following message:

```
Message type: Reply (7)
Transaction-ID: 0x001840f8
Client Identifier
    option type: 1
    option length: 14
    DUID type: link-layer address plus time (1)
    Hardware type: 0
    Time: 1099494430
    Link-layer address
Domain Search List
    option type: 24
    option length: 31
    DNS Domain Search List
Server Identifier
    option type: 2
    option length: 14
    DUID type: link-layer address plus time (1)
    Hardware type: 0
    Time: 1099923422
    Link-layer address
```

- ***Linux***

Client sends *Information-request* containing option Domain Search List. The message is complaint with RFC 3315 [RFC3315] and RFC 3736 [RFC3736]. Server responds to the client with *Reply* message containing domain search list. Client updates the domain search list in the resolv.conf, but also adds *0xCD* character at the end of the domain search list before the *CR* character. It is only the case when only one domain search list is returned.

- ***Windows***

Not implemented, no actions are taken.

5.3.5.2.5 Reliability of client initiated message exchanges test

This test case tests the retransmission strategy to be used by clients in client-initiated message exchanges. With each message transmission or retransmission, the client sets

Retransmission Timeout. If retransmission time expires before the message terminates, the client recomputes the retransmission time and retransmits the message.

- ***Status***

Test failed.

The series of Retransmission Timeouts are not described by the RT equation in RFC 3315 [RFC3315].

- ***Preconditions***

No connection with dhcpv6 server. Client sent *Information-request* message.

- ***Input***

client.conf

```
log-mode short
iface eth0
{
    stateless
    option domain
    T1 60
    T2 60
}
```

- ***Expected output***

Client should wait retransmission timeout equal to *RT = 2 * RT prev + RANDOM * RT prev* where:
RT prev is previous retransmission timeout. *RANDOM* is s randomization factor, which is a random number with a uniform distribution between -0.1 and +0.1.

The algorithm for choosing random number should produce a different sequence of random numbers from each invocation of the DHCP client.

Maximum retransmission count, maximum retransmission time and maximum retransmission duration are not implemented in Dibbler client, so they are not taken into account.

- ***Actual output***

Client sends correct *Information-request* messages after the retransmission timeout.
The following retransmission timeouts were obtained:

1, 2, 4, 8, 16, 33, 64, 131, 122, 131, 128, 117, 119, 116, 113, 115, 119, 116, 122, 123, 132, 121, 121

The series above are not described by the RT equation in RFC 3315 [RFC3315].

5.3.5.3 Test results for Dibbler server

Server was lunched on Linux and Windows platform and some unexpected behavior was noticed.

Format of all messages sent by server was checked against RFC 3315 [RFC3315].

5.3.5.3.1 Stateless server test

Check if server acts as a stateless server. Client asks for address and DNS name server. Stateless server should response with DNS server name only.

- ***Status***

Test failed.

Server sends every information client asked. This behavior is wrong since RFC 3315 [RFC3315] says that stateless servers cannot send any addresses.

- ***Expected output***

Client initiates process to gain IPv6 address and DNS server name. Stateless server should response with DNS server name only.

- ***Actual output***

Message sent to server:

```
Message type: Request (3)
    Transaction-ID: 0x000042db
    Client Identifier
        option type: 1
        option length: 14
        DUID type: link-layer address plus time (1)
        Hardware type: 6
        Time: 1100075912
        Link-layer address
    Identify Association
        option type: 3
        option length: 40
        IAID: 2
        T1: 4294967295
        T2: 4294967295
        IA Address
            option type: 5
            option length: 24
            IPv6 address: ::
            Preferred lifetime: infinity
            Valid lifetime: infinity
    Elapsed time
        option type: 8
        option length: 2
        elapsed-time: 3 sec
    Option Request
        option type: 6
        option length: 2
        Requested Option code: DNS recursive name server (23)
    Server Identifier
        option type: 2
```

```
        option length: 14
        DUID type: link-layer address plus time (1)
        Hardware type: 0


        Time: 1100077688
        Link-layer address
```

Message sent from server to client:

```
Message type: Reply (7)
    Transaction-ID: 0x000042db
    Client Identifier
        option type: 1
        option length: 14
        DUID type: link-layer address plus time (1)
        Hardware type: 6
        Time: 1100075912
        Link-layer address
    Identify Association
        option type: 3
        option length: 74
        IAID: 2
        T1: 60
        T2: 60
        IA Address
            option type: 5
            option length: 24
            IPv6 address: 2001::2e
            Preferred lifetime: 1800
            Valid lifetime: 3600
        Status code
            option type: 13
            option length: 30
            Status Code: Success (0)
            Status Message: All addresses were assigned.
    Server Identifier
        option type: 2
        option length: 14
        DUID type: link-layer address plus time (1)
        Hardware type: 0
        Time: 1100077688
        Link-layer address
    DNS recursive name server
        option type: 23
```

```
option length: 32
DNS servers address: 2000::500
DNS servers address: 2000::501
```

5.3.5.3.2 SIP Server test

Check if SIP server option is supported.

- ***Status***

Test passed.

Client wants to get DNS server name and SIP server. Server returns message with information client asked for.

- ***Preconditions***

Both clients and server interact for the first time.

- ***Input***

client.conf:

```
log-level 8
log-mode short

iface eth0
{
      option dns-server
      option sip-server
      ia {
      }
}
```

server.conf:

```
iface eth0
{
       option     dns-server        2000::600, 2000::601
       option     domain            test.com
       option     sip-server        2000::99

       class {
                     pool 2000::1-2000::ff
       }
}
```

- ***Expected output***

Client needs to get DNS server name and SIP server. Upon receiving request server should provide client with that information.

- ***Actual output***

Message sent to server:

```
 Message type: Information-request (11)
    Transaction-ID: 0x0000247b
    Client Identifier
        option type: 1
        option length: 14
        DUID type: link-layer address plus time (1)
        Hardware type: 6
        Time: 1100075912
        Link-layer address
    Elapsed time
        option type: 8
        option length: 2
        elapsed-time: 1 sec
    Option Request
        option type: 6
        option length: 6
        Requested Option code: DNS recursive name server (23)
        Requested Option code: SIP Server Domain Name List (21)
        Requested Option code: Unknown (42)
```

Message sent back from server to client:

```
Message type: Reply (7)
    Transaction-ID: 0x0000247b
    Client Identifier
        option type: 1
        option length: 14
        DUID type: link-layer address plus time (1)
        Hardware type: 6
        Time: 1100075912
        Link-layer address
    DNS recursive name server
        option type: 23
        option length: 32
        DNS servers address: 2000::500
        DNS servers address: 2000::501
    SIP Server Domain Name List
```

```
            option type: 21
            option length: 16
            SIP Servers Domain Search List
            SIP servers address: 2000::99
        Server Identifier
            option type: 2
            option length: 14
            DUID type: link-layer address plus time (1)
            Hardware type: 0
            Time: 1100077688
            Link-layer address
```

5.3.5.3.3 Accept only certain clients test

Check to see if server is able to accept only clients from specific address range.

- ***Status***

Test passed.

Server performs message sending only to the clients who are on the accept list. Other clients are denied.

- ***Preconditions***

Both clients and server interact for the first time.

- ***Input***

server.conf:

```
log-level 8
log-mode short

iface eth0
{
      option            dns-server        2000::600, 2000::601
      option      domain            test.com

       class {

             accept-only 2000::00-2000::20

               pool 2000::1-2000::ff
       }
}
```

- ***Expected output***

Clients send periodically *Information-request* message in order to get some configuration parameters. Server should deny those clients who are not on the accept list.

- ***Actual Output***

Server performs message sending only to the clients who are on the accept list. Other clients are denied.

5.3.5.3.4 Reject certain clients test

This test checks if server has ability to refuse clients specified on the reject-clients list.

- ***Status***

Test passed.

Server sends messages only to the clients that are out of the reject-clients list range. Clients with rejected addresses receive no messages.

- ***Preconditions***

Both clients and server interact for the first time.

- ***Input***

server.conf:

```
log-level 8
log-mode short

iface eth0
{
       option     dns-server        2000::600, 2000::601
       option     domain            test.com

       class {
                      pool 2000::1-2000::ff
       }
}
```

- ***Expected output***

Clients send periodically INFORMATION-REQUEST message in order to get some configuration parameters. Server should deny clients message whose address is within reject list range.

- ***Actual Output***

Server sends messages only to the clients that are out of the reject-clients list range. Clients with rejected addresses receive no messages.

5.3.5.4 Conclusions

Both server and client are configurable through their configuration file, which has its own text format. Although format is simple, it is not portable. We suggest that it should be written as an xml file.

Many features are not implemented in Dibbler [Dibbler]. The full comparison is shown in table 6. Some features are described in Dibbler's documentation, others were discovered during tests. Dibbler suffers from lack of authentication. Although this option is optional it could greatly improve safety of sending of messages and prevent from possible attacks.

Table 6. Features in Dibbler

Feature	Status
Transmission of messages	Implemented
Reliability (retransmission)	Implemented
Preference	Not implemented
User Class	Not implemented
Vendor Class	Not implemented
Vendor-specific Information	Not implemented
Authentication	Not implemented
DNS servers	Implemented
Domain search list	Implemented, but Windows version does not interact with operating system environment
SIP servers	Implemented, but neither Linux port nor Windows port set it in operating system environment
Elapsed time	Implemented
Client identifier	Implemented
Relay agents	Not implemented
Server identifier	Implemented

Both server and client proved to be stable. During tests, which some of them took several hours, no crash occurred. Tests were performed under Linux and Windows. Test results are shown in table 7.

Structure of messages is the same as RFC 3315 [RFC3315] and RFC 3736 [RFC3736] specification. What is interesting and might be a bug in client software is the fact that *Information-request* message sent by client contains unknown option (42) when *Information-request* message contains any query option.

Additionally, Dibbler server cannot be configured to run only in stateless mode. Dibbler server replies to both stateless and stateful clients although RFC 3736 [RFC3736] states that stateless server should discard any stateful messages.

The Retransmission Timeout series resulting from client are not compliant to RFC3315 [RFC3315], because they are not described by the RT equation in RFC3315 [RFC3315].

Table 7. Test results.

Test goal	Result
Client message format	Message structure is RFC3315 [RFC3315] and RFC3736 [RFC3736] conformant, but every *Information-request* message contains unknown option (42).
Transmission of messages	• Client uses multicast to send messages. • Client sends only *Information-Request* messages to server. • Client can send *Information-Request* asking for DNS servers, DNS search domains and SIP servers. • **Client cannot send correct *Information-Request* message after receiving *Reply* message from server.** • Client does not implement *Preference* option, *User Class* option, *Vendor Class* option, *Vendor-specific Information* option • Server can reply for DNS servers, DNS search domains and SIP servers requests from client. • **Server not only responds with *Reply* messages to client.** • Server implements *Client Identifier* option. • Server does not implement *Preference* option, *Vendor-specific Information* option. • Server does not implement communication with relay agents.
Reliability of client initiated message exchanges	• **The Retransmission Timeout series resulting from client are not described by the RT equation in RFC 3315 [RFC3315] (not compliant).**
Authentication	• Authentication is not implemented in client and server.

Test goal	Result
Operating system interaction	• **Client not always sets DNS servers received from the server correctly in the operating system environment in Linux environment.** • Client can set DNS servers received from the server in the operating system environment in Windows XP environment. • **Client not always sets domain search list received from the server correctly in the Linux operating system environment.** • **Client cannot set domain search list received from the server in the Windows XP operating system environment – not implemented** • **Client cannot set SIP servers list received from the server in the operating system environment.** • **Client cannot clean-up the system when client is shut down (killed) on Linux environment** • Client can clean-up the system when client is shut down (killed) on Windows XP environment

Dibbler client, after receiving *Reply* message from server, sends every *Information-request* message, which does not contain any option although it should contain at least one. Therefore server does not respond to client's requests. This is correct behaviour for server but incorrect for client. This is also the case for domain search list and sip servers list options.

Client not always sets DNS servers received from the server correctly in the Linux environment. If resolv.conf doesn't end with *CR* the resulting resolv.conf is corrupted, because Dibbler does not append *CR* before appending dns-server list.

On Linux client updates the domain search list in the resolv.conf, but also adds *0xCD* character at the end of the domain search list before the *CR* character. It is only the case only when returned search list contains one domain.
Windows XP client does not implement two important features. It cannot set domain search list received from the server in the operating system environment. This is also the case for SIP servers list. SIP servers are not also implemented in Linux port of Dibbler. Returned SIP servers are stored in dibbler's xml files.

When client is shut down on Linux it removes from resolv.conf DNS servers and domain search lists received from server. Unfortunately, it also changes the resolv.conf's access control list to *0014*. Although removal of DNS servers and domain search lists received from server is desired, access control list should not be changed.

It is also interesting to note, although it was not the goal of this document, that according to Dibbler's author [Dibbler] core logic is system independent, which is not true for client implementation. It was revealed accidentally by detecting the bug that we stuck into while testing Linux client. There are fragments in core logic that are operating system dependent. List of network interfaces is not reset and pointer does not point to the first interface on the list. It is a case for non-Windows ports.

Taking into consideration all test results that were made it is not recommended to use Dibbler server and client in production environment.

5.4 HP-UX

The HP-UX DHCPv6 implementation is available for HP-UX only. So far none of the 6NET partners has an HP-UX system at hand and is ready to test this implementation.

5.5 KAME

5.5.1 Description

This implementation supports prefix delegation, DNS server updates which feature is extensively used some 6NET partners, list of NTP servers, list of SIP servers but those features are not validated yet. It also supports relaying, however it is not tested.

5.5.2 Documentation

Documentation is provided by KAME as man pages. The following manuals are available dhcp6c.conf(5), dhcp6s.conf(5), dhcp6c(8), dhcp6s(8), dhcp6relay(8), dhcp6sctl(8).

5.5.3 Functionality

The authors of the software provide the following feature list:

- IPv6 prefix delegation
- List of the DNS server addresses
- DNS Domain search list
- List of the NTP server addresses
- List of the SIP server addresses
- SIP server domain
- Supported Options: Rapid commit, Server Preference, Information Request.
- Elapsed Time, ClientID, ServerID, IA_PD, Status support.
- Solicit/Request/Advertise/Reply/Information-request messages/Renew/Rebind/Release/Confirm/ messages support for IPv6 address binding.
- Client configuration file support for IPv6 prefix delegation
- Prefix delegation support according to draft-ietf-dhc-dhcpv6-opt-prefix-delegation-03.txt

ToDo List

- Authentication/User class/Vendor class/Interface-ID option support
- Relay agent support.
- Reconfig/Relay messages support.

5.5.4 Test of stateless operation

5.5.4.1 Environment

The tests are executed on FreeBSD (4.10-STABLE and 5.3-STABLE). The software should be easily ported to other BSD platforms and Linux also.

5.5.4.2 Compiling the code/ Installing the software

This is how it is described and it works fine:

1. cd /usr/ports/net/dhcp6/
2. make install ; make clean
3. cd /usr/local/etc
4. cp dhcp6c.conf.sample dhcp6c.conf (copy client configuration file)
5. cp dhcp6s.conf .sample dhcp6s.conf (copy server configuration file)
6. /usr/local/sbin/dhcp6s -Df [fxp0 bge1 ...] (start server, turn on debug)
7. /usr/local/sbin/dhcp6c -Df fxp0 (start client, turn on debug)

5.5.4.3 Tests

We tested DNS server configuration via DHCPv6 on the same link with two different setups:

- DNS server configuration provided by KAME DHCPv6 server and used by KAME DHCPv6 client.
- DNS server configuration provided by Cisco DHCPv6 server and used by KAME DHCPv6 client
-

5.5.4.3.1 KAME DHCPv6 server

Client and server worked fine when using configuration built up by information provided by the man pages.

Server configuration file:

```
option domain-name-servers 2001:db8::35;
```

Client config:

```
Interface fxp0{
     information-only;
};
```

5.5.4.3.1.1 Client debug output

```
dhcp6c -Df fxp0
Nov/17/2004 23:16:48: cfdebug_print: <3>comment [# The followings are a sample configuration for requiring the "stateless"] (73)
Nov/17/2004 23:16:48: cfdebug_print: <3>comment [# DHCPv6 service.] (17)
Nov/17/2004 23:16:48: cfdebug_print: <3>[interface] (9)
Nov/17/2004 23:16:48: cfdebug_print: <5>[fxp0] (4)
Nov/17/2004 23:16:48: cfdebug_print: <3>begin of closure [{] (1)
Nov/17/2004 23:16:48: cfdebug_print: <3>[information-only] (16)
```

```
Nov/17/2004 23:16:48: cfdebug_print: <3>end of sentence [;] (1)

Nov/17/2004 23:16:48: cfdebug_print: <3>end of closure [}] (1)

Nov/17/2004 23:16:48: cfdebug_print: <3>end of sentence [;] (1)

Nov/17/2004 23:16:48: get_duid: extracted an existing DUID from
/var/db/dhcp6c_duid: 00:01:00:01:09:2e:84:8c:36:4f:c0:b2:14:30

Nov/17/2004 23:16:48: dhcp6_reset_timer: reset a timer on fxp0, state=INIT,
timeo=0, retrans=383

Nov/17/2004 23:16:48: client6_send: a new XID (87592b) is generated

Nov/17/2004 23:16:48: copy_option: set client ID (len 14)

Nov/17/2004 23:16:48: copy_option: set elapsed time (len 2)

Nov/17/2004 23:16:48: client6_send: send information request to ff02::1:2%fxp0

Nov/17/2004 23:16:48: dhcp6_reset_timer: reset a timer on fxp0, state=INFOREQ,
timeo=0, retrans=988

Nov/17/2004 23:16:48: client6_recv: receive reply from
fe80::20f:1fff:fea4:ba0a%fxp0 on fxp0

Nov/17/2004 23:16:48: dhcp6_get_options: get DHCP option client ID, len 14

Nov/17/2004 23:16:48:   DUID: 00:01:00:01:09:2e:84:8c:36:4f:c0:b2:14:30

Nov/17/2004 23:16:48: dhcp6_get_options: get DHCP option server ID, len 14

Nov/17/2004 23:16:48:   DUID: 00:01:00:01:09:2e:84:4e:36:4f:c0:b2:14:30

Nov/17/2004 23:16:48: dhcp6_get_options: get DHCP option DNS, len 16

Nov/17/2004 23:16:48: client6_recvreply: nameserver[0] 2001:db8::35

Nov/17/2004 23:16:48: dhcp6_remove_event: removing an event on fxp0,
state=INFOREQ

Nov/17/2004 23:16:48: client6_recvreply: got an expected reply, sleeping.
```

5.5.4.3.1.2 Cisco debug output

```
dhcp6s -c /usr/local/etc/dhcp6s.conf -Df fxp0

Nov/17/2004 23:16:32: cfdebug_print: <3>comment [# The followings are a sample
configuration to provide a DNS server address] (75)

Nov/17/2004 23:16:32: cfdebug_print: <3>comment [# for every client as well as
to delegate a permanent IPv6 prefix] (65)

Nov/17/2004 23:16:32: cfdebug_print: <3>comment [# 2001:db8:1111::/48 to a
client whose DUID is 00:01:00:01:aa:bb.] (65)

Nov/17/2004 23:16:32: cfdebug_print: <3>[option] (6)

Nov/17/2004 23:16:32: cfdebug_print: <3>[domain-name-servers] (19)

Nov/17/2004 23:16:32: cfdebug_print: <3>[2001:db8::35] (12)

Nov/17/2004 23:16:32: cfdebug_print: <3>end of sentence [;] (1)

Nov/17/2004 23:16:32: get_duid: extracted an existing DUID from
/var/db/dhcp6s_duid: 00:01:00:01:09:2e:84:4e:36:4f:c0:b2:14:30

Nov/17/2004 23:16:32: ctlauthinit: failed to open /usr/local/etc/dhcp6sctlkey:
No such file or directory

Nov/17/2004 23:16:32: server6_init: failed initialize control message
authentication
```

```
Nov/17/2004 23:16:32: server6_init: skip opening control port
Nov/17/2004 23:16:48: server6_recv: received information request from fe80::20f:1fff:fea4:ba0a%fxp0
Nov/17/2004 23:16:48: dhcp6_get_options: get DHCP option client ID, len 14
Nov/17/2004 23:16:48:   DUID: 00:01:00:01:09:2e:84:8c:36:4f:c0:b2:14:30
Nov/17/2004 23:16:48: dhcp6_get_options: get DHCP option elapsed time, len 2
Nov/17/2004 23:16:48:   elapsed time: 0
Nov/17/2004 23:16:48: copy_option: set client ID (len 14)
Nov/17/2004 23:16:48: copy_option: set server ID (len 14)
Nov/17/2004 23:16:48: copy_option: set DNS (len 16)
Nov/17/2004 23:16:48: server6_send: transmit reply to fe80::20f:1fff:fea4:ba0a%fxp0
```

5.5.4.3.1.3 Assessment of test result

Both client and server properly formed their DUID. The client was able to request information from the server and report it back the configured DNS nameserver to dhcp6c client program.

5.5.4.3.2 Cisco DHCPv6 server with KAME DHCPv6 client

Client and server worked fine when using configuration built up by information provided by the man pages.

Cisco router configuration snippets:

```
ipv6 dhcp pool dhcp6dns
       dns-server 2001:738:0:402::2
       domain-name ki.iif.hu
```

and on the interface configuration:

```
     ipv6 dhcp server dhcp6dns
```

Client config (same as before):

```
Interface fxp0{
     information-only;
};
```

5.5.4.3.2.1 Client debug output

```
/usr/local/sbin/dhcp6c -Df fxp0
Nov/18/2004 16:58:45: cfdebug_print: <3>comment [# The followings are a sample configuration for requiring the "stateless"] (73)
Nov/18/2004 16:58:45: cfdebug_print: <3>comment [# DHCPv6 service.] (17)
Nov/18/2004 16:58:45: cfdebug_print: <3>[interface] (9)
```

```
Nov/18/2004 16:58:45: cfdebug_print: <5>[fxp0] (4)
Nov/18/2004 16:58:45: cfdebug_print: <3>begin of closure [{] (1)
Nov/18/2004 16:58:45: cfdebug_print: <3>[information-only] (16)
Nov/18/2004 16:58:45: cfdebug_print: <3>end of sentence [;] (1)
Nov/18/2004 16:58:45: cfdebug_print: <3>end of closure [}] (1)
Nov/18/2004 16:58:45: cfdebug_print: <3>end of sentence [;] (1)
Nov/18/2004 16:58:45: get_duid: extracted an existing DUID from
/var/db/dhcp6c_duid: 00:01:00:01:09:2e:84:8c:36:4f:c0:b2:14:30
Nov/18/2004 16:58:45: dhcp6_reset_timer: reset a timer on fxp0, state=INIT,
timeo=0, retrans=383
Nov/18/2004 16:58:45: client6_send: a new XID (dcd7e) is generated
Nov/18/2004 16:58:45: copy_option: set client ID (len 14)
Nov/18/2004 16:58:45: copy_option: set elapsed time (len 2)
Nov/18/2004 16:58:45: client6_send: send information request to ff02::1:2%fxp0
Nov/18/2004 16:58:45: dhcp6_reset_timer: reset a timer on fxp0, state=INFOREQ,
timeo=0, retrans=988
Nov/18/2004 16:58:45: client6_recv: receive reply from
fe80::20b:45ff:fea4:f808%fxp0 on fxp0
Nov/18/2004 16:58:45: dhcp6_get_options: get DHCP option server ID, len 10
Nov/18/2004 16:58:45:   DUID: 00:03:00:01:00:0b:45:a4:f8:06
Nov/18/2004 16:58:45: dhcp6_get_options: get DHCP option client ID, len 14
Nov/18/2004 16:58:45:   DUID: 00:01:00:01:09:2e:84:8c:36:4f:c0:b2:14:30
Nov/18/2004 16:58:45: dhcp6_get_options: get DHCP option DNS, len 16
Nov/18/2004 16:58:45: dhcp6_get_options: get DHCP option domain search list, len
11
Nov/18/2004 16:58:45: client6_recvreply: nameserver[0] 2001:738:0:402::2
Nov/18/2004 16:58:45: client6_recvreply: Domain search list[0] ki.iif.hu.
Nov/18/2004 16:58:45: dhcp6_remove_event: removing an event on fxp0,
state=INFOREQ
Nov/18/2004 16:58:45: client6_recvreply: got an expected reply, sleeping.
```

5.5.4.3.2.2 Server debug output

```
cntrl.6net.hbone.hu#debug ipv6 dhcp detail

IPv6 DHCP debugging is on (detailed)

cntrl.6net.hbone.hu#

*Nov 18 15:48:51 UTC: IPv6 DHCP: Received INFORMATION-REQUEST from
FE80::20F:1FFF:FEA4:BA0A on GigabitEthernet0/0.801

*Nov 18 15:48:51 UTC: IPv6 DHCP: detailed packet contents

*Nov 18 15:48:51 UTC:   src FE80::20F:1FFF:FEA4:BA0A (GigabitEthernet0/0.801)

*Nov 18 15:48:51 UTC:   dst FF02::1:2

*Nov 18 15:48:51 UTC:   type INFORMATION-REQUEST(11), xid 904574

*Nov 18 15:48:51 UTC:   option CLIENTID(1), len 14

*Nov 18 15:48:51 UTC:     00010001092E848C364FC0B21430

*Nov 18 15:48:51 UTC:   option ELAPSED-TIME(8), len 2

*Nov 18 15:48:51 UTC:     elapsed-time 0

*Nov 18 15:48:51 UTC: IPv6 DHCP: Sending REPLY to FE80::20F:1FFF:FEA4:BA0A on
GigabitEthernet0/0.801

*Nov 18 15:48:51 UTC: IPv6 DHCP: detailed packet contents

*Nov 18 15:48:51 UTC:   src FE80::20B:45FF:FEA4:F808

*Nov 18 15:48:51 UTC:   dst FE80::20F:1FFF:FEA4:BA0A (GigabitEthernet0/0.801)
```

```
*Nov 18 15:48:51 UTC:   type REPLY(7), xid 904574
*Nov 18 15:48:51 UTC:   option SERVERID(2), len 10
*Nov 18 15:48:51 UTC:     00030001000B45A4F806
*Nov 18 15:48:51 UTC:   option CLIENTID(1), len 14
*Nov 18 15:48:51 UTC:     00010001092E848C364FC0B21430
*Nov 18 15:48:51 UTC:   option DNS-SERVERS(23), len 16
*Nov 18 15:48:51 UTC:     2001:738:0:402::2
*Nov 18 15:48:51 UTC:   option DOMAIN-LIST(24), len 11
*Nov 18 15:48:51 UTC:     ki.iif.hu
```

5.5.4.3.2.3 Assessment of test result

Both client and server properly formed their DUID. The client was able to request information from the server and report it back the configured DNS nameserver and domain-list to dhcp6c client program.

5.5.4.4 Conclusions

The KAME implementation can be used effectively for stateless operation to distribute DNS information for autoconfigured clients. We are using it in the production environment with Cisco router as a DHCPv6 server. We look forward to testing the relaying capabilities of both Cisco and KAME implementation to provide DNS configuration information for autoconfigured IPv6 clients.

5.6 Cisco

5.6.1 Relay

Since 12.3(11)T the Cisco IOS contains a dhcpv6 relay. JOIN tested it together with the NEC Netlabs DHCP server and Client. The relay behaves as it was expected. To enable the relaying on a link there is only one command for the appropriate interface necessary:

Router:(config-if)# ipv6 dhcp relay destination <ipv6-address-of-dhcpv6-server>

A short test was done as an interoperability test (see below in section 5.9).

5.7 Juniper

We are planning to test it for stateless operation to distribute DNS information.

5.8 Hitachi

So far not tested.

5.9 Interoperability

5.9.1 Example interoperability test

5.9.1.1 Environment

The Sourceforge DHCPv6 client v0.10 was tested on Linux together with the Cisco IOS stateless DHCP server (IOS 12.3(11.1)T).

5.9.1.2 Configuration

The sourceforge client uses wrong DNS option numbers while Cisco uses what is specified in RFC 3646. This can be fixed by configure by using the following configure flags:

```
--with-opt-dns-resolvers=23 --with-opt-domain-list=24
```

5.9.1.2.1 Client config

This enables the client for "stateless mode" and asks for DNS info:

```
interface eth0 {
        request domain-name-servers;
        information-only;
};
```

5.9.1.2.2 Server config 1

```
ipv6 dhcp pool uninett
 dns-server 2001:700:0:503::CA53
 domain-name uninett.no
```

5.9.1.2.3 Server config 2

Adding an additional nameserver:

```
ipv6 dhcp pool uninett
 dns-server 2001:700:0:503::CA53
 dns-server 2001:700:0:503::CA54
 domain-name uninett.no
```

5.9.1.2.4 Server config 3

Defining an IPv4 nameserver:

```
ipv6 dhcp pool uninett
 dns-server ::FFFF:158.38.60.10
```

```
domain-name uninett.no
```

That is, using IPv4-mapped IPv6 address as defined by RFC 3513.

5.9.1.3 Results

5.9.1.3.1 Server config 1

This basic configuration works as desired, creating the following resolv.conf on the client:

```
nameserver 2001:700:0:503::ca53
search uninett.no
```

5.9.1.3.2 Server config 2

In this case the client then created the resolv.conf

```
nameserver 2001:700:0:503::ca53 2001:700:0:503::ca54
search uninett.no
```

which is wrong. It should have used two nameserver lines.

5.9.1.3.3 Server config 3

Result is the resolv.conf file:

```
nameserver ::ffff:158.38.60.10
earch uninett.no
```

Which indeed worked fine with the glibc resolver on Linux. That is, it allows mapped addresses, and used IPv4 with the address 158.38.60.10. Configuring IPv4 servers (or a mix of v4 and v6 servers) with DHCPv6 in this way seems to be a bit controversial and may not work, but it might be useful.

5.9.2 Interoperability matrix for stateful operation

After initial tests described in 5.1 to 5.3 the interoperability of the tested implementations was checked. With respect to the fact that neither sourceforge.net nor dibbler version supports relaying there will be only a matrix for tests on a single link.

Server Client	NEC	Dibbler	Sourceforge.net

NEC	Works	Works with minor problems	Fails
Dibbler	Works	Normally works but there are minor problems	Fails
Sourceforge.net	Works	Fails	Works

The minor problems as described in the dibbler tests section are the assignment of ::/128 to the configured interface. During testing the sourceforge.net server with various clients the server dies after getting the configuration request sent by a NEC-client and together with the dibbler-client it was not possible to get an address configured on the given interface, because of problems regarding dhcp options. Using the dibbler server the sourceforge client dies at receiving the configuration data and the NEC client worked but set ::/128 as an additional address to the configured interface.

6 Conclusion

Speaking of address delegation with DHCPv6, the implementations mostly seem to be immature. Especially, most implementations lack DHCPv6 relay support, which is a vital part to deploy DHCPv6 in a larger network. In addition, the implementations do not interoperate properly. Obviously there has to take place a lot of more interaction between the different developers. So from a today's point of view, stateful DHCPv6 service is not deployable or at least only with some limitations.

The stateless mode of DHCPv6 and especially the prefix delegation and nameservice option are more widely implemented as they seem to be more important for today's networks. This document does not show detailed tests of all implementations for these features, but to the knowledge of the authors, prefix delegation and nameservice option should work fine for most of them. The lack of DHCPv6 relays is not so important in the stateless mode, as one could set up a DHCPv6 server on every router link. So at least the stateless feature sets seem to be deployable today. Nevertheless, this still has to be confirmed and tested more thoroughly.

References

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

[RFC2462] "IPv6 Stateless Address Autoconfiguration", S. Thomson, T. Narten, RFC 2462; December 1998.

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

[RFC3736] "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", R. Droms, RFC 3736; April 2004.

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

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

[RFC3319] "Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers", H. Schulzrinne, B. Volz, RFC 3319; July 2003.

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

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

[RFC3898] "Network Information Service (NIS) Configuration Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", V. Kalusivalingam, RFC 3898; October 2004.

[SNTP] "Simple Network Time Protocol Configuration Option for DHCPv6", A.K. Vijayabhaskar, Internet Draft, draft-ietf-dhc-dhcpv6-opt-sntp-00.txt; November 2003.

[LIFETIME] "Information Refresh Time Option for DHCPv6", S. Venaas, T. Chown, B. Volz, Internet Draft, draft-ietf-dhc-lifetime-02.txt; September 2004.

[CLIENTFQDN] "The DHCPv6 Client FQDN Option", B. Volz, Internet Draft, draft-ietf-dhc-dhcpv6-fqdn-00.txt; September 2004.

[DSTM] "DSTM Options for DHCP", B. Volz, J. Bound, R. Droms, T. Lemon, Internet Draft (expired), draft-ietf-dhc-dhcpv6-opt-dstm-01.txt; April 2002.

[DSTMPORT] "DSTM Port Options for DHCP", M.-K. Shin, Internet Draft (expired), draft-ietf-dhc-dhcpv6-opt-dstm-ports-01.txt; June 2002.

[CLIPREF] "Client Preferred Prefix option for DHCPv6", A.K. Vijayabhaskar, Internet Draft (expired), draft-ietf-dhc-dhcpv6-opt-cliprefprefix-01.txt; March 2003.

[LOADB] "Load Balancing for DHCPv6", B. Volz, Internet Draft (expired), draft-ietf-dhc-dhcpv6-loadb-02.txt; August 2002.

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

[RFC3118] "Authentication for DHCP Messages", R. Droms (Ed.), W. Arbaugh (Ed.), RFC 3118; June 2001.

[D3.2.3v1] 6NET Deliverable 3.2.3: "DHCPv6 implementation and test report" (first version); January 2002.

[Dibbler] Mrugalski T., “Dibbler – a portable DHCPv6 User's guide version 0.2.1-RC1”, http://www.klub.com.pl/dhcpv6, October 2004.

[Ethereal] Combs G., Ethereal 0.10.5, http://www.ethereal.com/introduction.html, 2004.