Next Generation Automatic IP Configuration Deployment Issues

Tomasz Mrugalski
Gdansk University of Technology, Intel Corp.
tomasz.mrugalski@intel.com

Krzysztof Nowicki
Gdansk University of Technology
know@eti.pg.gda.pl,

Krzysztof Wnuk
Gdansk University of Technology,
keczi@kastor.ds.pg.gda.pl

Abstract

Although Dynamic Host Configuration Protocol for IPv6 (DHCPv6) protocol was defined in 2003, it was designed as a framework rather than a complete solution to the automatic configuration in IPv6 networks. There are still some unsolved problems and new options yet to be defined. One example of such case is Fully Qualified Domain Name (FQDN) option, which final version has been published in late 2007. It describes DHCPv6 client and server behavior, but some important aspects remain unaddressed. Authors developed and released working open source implementation over a year before FQDN standard reached mature phase. This paper discuses those issues and recommends possible solutions. Another important development area in the DHCPv6 protocol is a lack of well defined authentication and authorization. Experimental AAA implementation has been developed and reached validation phase. Quick overview of the Dibbler project – a working, multi-platform, open-source DHCPv6 implementation – is also provided, its strengths, used solutions and validation methods used to prove its correctness are discussed. Conclusions and discussion regarding areas for further studies appears in the last section of this article.

1. Introduction

DHCPv6 design was designed as a framework, which provides easy way for adding new options and features. Due to unified option handling, it is possible to maintain both forward and backward compatibility, i.e. implementation supporting only base standard during cooperation can safely ignore not supported options and continue message parsing. DHCPv6 as a protocol ensure obtaining IPv6 address and other configuration parameters in both stateful and stateless mode. When stateful mode is used, DHCPv6 grants one or more address to the client. Besides just an address, various additional options can be provided. One of them is a Fully Qualified Domain Name (FQDN). When client receives its hostname, it can be used locally only (e.g. for logging purposes). To allow other nodes to know about granted addresses and hostname, a DNS server has to be updated. There are two DNS Updates possible: forward (hostname to address resolution, so called AAAA record) and reverse (address to hostname, so called PTR record) entries. FQDN Option was designed to provide client with a possibility to update his AAAA record [5] in DNS server. This mechanism, which was not present in DHCP for IPv4 protocol, is a considerable challenge for developers, administrator and, to some lesser extent, end users.

Another significant issue is a cryptographic protection of the DHCPv6 messages. [2] proposes usage of IPSec for server-relay and relay-relay exchanges. Currently, there are no known implementations available that supports such features[1]. [2] merely specifies that “The client's keys are initially distributed to the client through some out-of-band mechanism. Mechanisms for key distribution and lifetime specification are beyond the scope of this document.” Two example protection methods are defined: trivial replay detection and a delayed Authentication Protocol. Due to lack of numerous necessary details, these options are considered a framework that separate documents will base upon. One of such documents is [6]. Extra options are specified so DHCPv6 messages can be used to create DHCPv6 Security Associations between the client and server.

2. Client FQDN Option usage

According to [1], Client FQDN option can be defined as follows. There is a client, which is interested in obtaining a valid hostname, which he can be referred to by his peers. To perform this, during communication with the DHCPv6 server, client sends Client FQDN Option, with proposed hostname and bit field denoting if client wants to perform DNS Update by itself or rather should server perform it for the client. As with every option sent by client, those values are treated only as hints for the server. Server can take them into consideration, but can also ignore them completely. After receiving option request, DHCPv6 server is checking his FQDN policy and generating response for a client, (setting bit field according to the configured DNS update policy). This mechanism is described in the Fig. 1 below. Proper support for the FQDN option includes also message exchange with third party (a DNS server supporting DNS Updates mechanism). At least two separate communication sessions might be performed: to update forward resolving (AAAA record) and reverse resolving (PTR record). There are several possible approaches to this problem.

[1] One of the authors, participated in 3 consecutive DHCPv6 interoperability events, held before IETF meetings, in Amsterdam, Vanchouver and Philadelphia. None of the participants provided full authentication capable DHCPv6 implementation.

3. Issues

As mentioned before, FQDN paper [1] does not address several important issues. This section discusses more important problems encountered during design, coding and testing phases of the Dibbler implementation and its FQDN support.

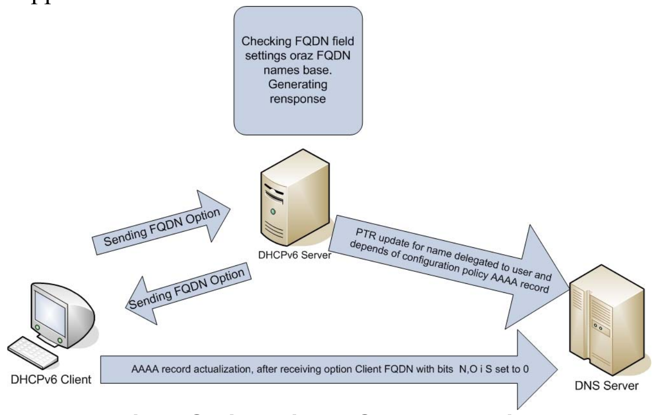
Checking FQDN field settings oraz FQDN names base. Generating renspone
Sending FQDN Option
DHCPv6 Server
PTR update for name delegated to user and depends of configuration policy AAAA record
Sending FQDN Option
DHCPv6 Client
AAAA record actualization, after receiving option Client FQDN with bits N,O i S set to 0
DNS Server

Fig. 1 Option client FQDN mechanism

3.1. Client-related issues

Client interested in getting FQDN name should announce this by including Client FQDN option in all transmitted messages. There are no rules that guarantees that domain name information will remain stable between messages. In particular if client has sent option to the server and then his configuration changed, he can send new option Client FQDN. This can be caused by a client manual reconfiguration not related to the DHCPv6. Another issue is related to the update responsibility. Client, who delegates update responsibility to the DHCPv6 server, will not receive any confirmation from the server that this update has been finished successfully. Finally client should perform record cleanup, if he is releasing address before lease time expires and he was responsible for AAAA update. If it fails while performing cleanup, this should be logged and reported to the user and site administrator. Failing to do so, will leave DNS server database in a disorder.

3.2. Server behavior

Server can include option Client Option FQDN only in ADVERTISE and REPLY messages, assuming that client included this option in previously sent message. After receiving FQDN option, server checks its configuration and update policy and generates response to the client (setting bits according to situation). Server configuration may allow using FQDN name provided by client or refusing it and using one from the pool configured on the server by network administrator.

Server is not allowed to start DNS update during ADVERTISE message generation, but it is possible to finish DNS update before sending REPLY message. As an alternative, server can send REPLY message to the client without waiting for the end of the DNS update process. If update process has not finished before server sends a response to the client, server may be forced to change domain name field for the client. This situation may occur when server discovers domain name conflict. In such case domain name chosen by the server will have to be changed. Server can also delay sending DHCP reply till DNS Update is completed. This poses a threat to propagate DNS update-related delays to the DHCPv4. If the delay is too large, this can trigger clients to retransmit its messages, which should not happen during normal operation. After receiving RELEASE or DECLINE message server should delete PTR record and according to update policy also AAAA record.

3.3. TTL for DNS records

DNS records connected with DHCPv6 client may be more dynamic than others. TTL value for RR record should be 1/3 life time address and should be more than 10 minutes. Of course setting this parameter depends on network administrator. Generally this value should be less that addresses life time. [1]

3.4. DNS update conflicts

[1] does not address this issue; it merely states that implementation should provide some mechanisms and take into consideration all possible situations. It is possible that during DNS Update attempt, entity performing update will discover that such name (or reverse pointer in case of reverse update) is already present. It might have been set by administrator, previous DHCP instance, which did not shutdown properly or simply exist as a configuration mistake. Using of security token may be implemented also (conflict may occur only in multiple updates at one time).

3.5. Security vs scalability

Unauthorized DNS updates or attacks may be a significant problem in DNS system. Therefore network administrators should have a full knowledge about clients, who will be allowed to perform DNS updates. Implementations may provide some mechanisms to increase updates security, but [1] doesn't describe any guidelines or hints. In all cases, the only entity allowed to perform PTR record update is a DHCPv6 server. Since the DHCPv6 server is configured in a manual (i.e. static) manner, the update clearance can be limited to its IPv6 address. Additional mechanisms like DHCPv6 authentication may be implemented to ensure security and avoid from sniffing and spoofing [7]. Also static DNSSec [11] keys usage can be configured. Since server can be configured to forbid any updates for clients and perform all updates by itself, this is the most secure approach.

However, this configuration scales poorly. Server has more tasks to execute with each new client. Since DHCPv6 failover and load balancing methods has not been defined yet, it is safe to assume that in large networks one DHCPv6 server will service large number of clients probably using several links – both local and remote (using relays). Executing DNS Update for each client from a single node (DHCPv6 server) might quickly become a bottleneck of the whole system. Therefore, from the scalability point of view, it is better to allow clients to perform their DNS Updates on their own, rather then configuring server to do it for clients.

Decision, if security or scalability is more important, is up to the network administrator.

3.6. Names reservation and management

Important part of the FQDN mechanism was a name reservation and management system. Several requirements for such system have been defined:

To meet all those requirements, text based list with optional reservations has been designed. For client identification purposes the natural choice was DUID – a DHCP Unique Identifier, a basic feature of the DHCPv6 protocol. To maintain name–address pair, IPv6 address has been chosen.

4. Implementation and validation

As a base for this implementation Dibbler [4] was used. Support for FQDN option as well as DNS Update mechanism has been designed, coded and tested. Results have been merged with the existing Dibbler code base and have been incorporated in the latest 0.5.0 release of the Dibbler. Dibbler as well as selected scenarios is described in the following subsections.

Dibbler is an open source, portable DHCPv6 implementation. It supports both stateful (i.e. IPv6 address granting) as well as stateless (i.e. option granting) autoconfiguration for IPv6. It features all DHCPv6 system components: server, client and relay. Supported platforms are Windows 2000, NT, XP and 2003. Linux version supports kernels 2.4 and 2.6. It is being distributed as an open source code, under GPL licence [4].

Dibbler project was started in 2003 and during its 3 year development, it has been accepted in several major Linux distributions (Debian, Gentoo, PLD, Ubuntu and others). Authors received feedback from users from 29 countries.

4.1. Working scenarios

All working scenarios – updates performed by client and server, and performed by server only – are fully supported and well tested. To setup DNS Update policy, fqdnMode parameter has been introduced.

4.2. Cleaning up record from DNS server

Cleaning up performs in the same way as adding new record. Side that was responsible for DNS update is deleting record. Authors chose to not provide possibility of deleting all RRsets for given name or deleting RRset[8]. This is the best solution from security point of view because we are not changing records that were not added by ourselves.

5. Authentication and Authorization

Selected security features have been implemented in the Dibbler code. Those are considered extra enhancements, not mentioned in the base specification [2], but proposed in [1]. They consist of extra options, which improves client—server communication by providing two extra properties: authorization and authentication. It eliminates threat of man-in-the-middle attacks. Such attacks could prove significant threat to the server, e.g. DNS spoofing. Additionally, client is being authorized during communication with server, so better control over clients' identity is provided.

Security improvements provided by authenticated DHCPv6 messages allow verification that received message was indeed transmitted by its supposed source. This is an essential feat for clients. This makes man-in-the-middle attacks much more difficult as attacked would have to compromise server before attacking clients. On the other hand, server can confirm client's identity. Without authentication, client spoofing is trivial – attacker has to sniff only one message from the client to gain all necessary knowledge (i.e. MAC address and a client DUID). Authentication support opens a way to implement reconfiguration. That is a mechanism specified in [2] used by servers to inform clients that new configuration is available and that clients should recheck their already leased parameters.

Typical authentication deployment scenario is as follows:

1. System administrator decides, which digest mode to use and generates shared key.
2. Copy of the keys is stored on the server and delivered to the client, using out-of-bond means.
3. Client begins operation. It sends SOLICIT message with a KEYGEN option and requests server to provide KEYGEN and AUTH options.
4. Server calculates the key required for digest generation. Following formula is used:

   key = HMAC-SHA-1 (AAA-key, {kgn || client-id}) (1)

   where kgn (Key Generation Nounce) is a at least 128 bit random sequence, generated by the server and sent to the client in the keygen option.
5. Server responds using ADVERTISE message that includes Key Generation option with SPI set to the value provided by the client. Additionally, this message is being signed using AUTH option.
6. Client receives SPI and stores it for further use. It also uses it, together with kng to generate key, which will be used in the AUTH option.
7. Client generates digest for received ADVERTISE message and verifies that it is identical with the value sent by the server. Message is dropped if any differences are detected.
8. Further messages sent between server and client are signed using AUTH option. Normal REQUEST-REPLY, RENEW-REPLY and RELEASE-REPLY exchanges are performed.

Details of the implementation are specified in [7].

6. Summary

Implementation of Option Client FQDN is world's first implementation in the open source community. Since it was based on internet draft and not a final specification, some behavior according to this option was not described in [1] as well as the way of updating DNS server. That is why authors solve these problems in following way:

- Update policy: How to define and implement DNS

Update policy from the server's point of view.

- Access control: How to organize and maintain access policy for allowing or denying clients' requests.
- Policy recommendations: Which update policy is recommended for various purposes?
- Negative scenarios: What DHCPv6 server or client should do in case of DNS update failure.
- DNS Update: How to perform DNS update (how to work with DNS resource records).
- Server only update – After receiving client's request, server checks update policy and sends back message with bit S set to 1. This denotes that client should skip DNS update. After sending REPLY message to the client, server performs both PTR and AAAA updates.
- Server is not performing DNS update at all – According to the draft [1], this scenario is also possible but rare. After receiving client's request for FQDN name, server sends response with bit N in bit fields set to 1 and does not perform any DNS update at all.
- Easy to configure simple approach, i.e. when administrator wants to configure only set of available domain names and don't care, which client gets which name.
- Name reservation for a particular client. There must be a way to reserve a specific name for a specific client. In this way, the same client will always get the same name.
- Constant Name–IPv6 address pair. It must be possible to maintain constant name with related IPv6 address, so every time the name is assigned to a client, also the same address is used.
- To provide secure environment, default update policy was delegated to DHCPv6 server. This approach allows restricting updates performed in the whole network to one node only. Reducing communication between DHCPv6 and DNS to one server is a obvious security improvement. Clients are not allowed to update FQDN in DNS server in this case.
- Adding new FQDN names and addresses take place in DHCPv6 server's configuration file. Administrator has possibility to define names, that will be reserved for particular client or DUID [2] (DUID is the best solution from maintenance perspective as it is consistent between reboots and even hardware changes) or free names, which can be delegated to any requesting client. Client's hint in this case is less important than server settings.
- While DNS update take place, only record directly connected with particular name or address are added or removed as it is described in section 2.5.4 [8]. Implementation does not delete any other DNS records, which were already present.
- Failover mode has been implemented. In case of DNS Update procedure fails, client FQDN Option for particular client is set as DISABLE and is skipped during next FQDN operation. Such approach solves case, when failing or misconfigured DNS server can cause DHCPv6 server to fail.

7. Further research

There are also some problems that need further investigation, research and eventually improvements:

- Implementation is based on AXFR zone transfer mechanism [9], which is not a optimal solution from a network traffic point of view. Initial research indicates that IXFR transfers should offer reduced network usage[10].
- Implementation uses connection oriented (TCP based) communication with the DNS server. Using datagram oriented (UDP based) approach might be more effective, but will not provide the same level of reliability.
- When performing reverse updates, Dibbler implementation does not support IP6.INT domain. This approach is obsolete, but due to backward compatibility support for this option might be taken into consideration.
- Convergence between DNS Update and Authentication. DNS Updates may also be performed using secure methods. That will be crucial, when DNS updates are to be performed by clients.

8. References

[1] B. Volz, „ *The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option*", RFC4704, IETF, October, 2006

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

[4] *Dibbler's home page* – http://klub.com.pl/dhcpv6/, retrieved March 2008

[5] S. Thomson, C. Huitema, V. Ksinant, M. Souissi "*DNS Extensions to Support IP Version 6*", RFC 3596, October 2003

[6] V.Ram, V.Kamble, S.Upadhyaya and N.Jain "*Authentication, Authorization and key management for DHCPv6*", IETF, work in progress, August 2006

[7] Michał Kowalczuk, "*DHCPv6 extenstion implementation for Linux and Windows systems*", master thesis, Gdansk University of Technology, December 2007.

[8] P. Vixie, S. Thomson, Y. Rekhter, J. Bound "*Dynamic Updates in the Domain Name System*", RFC 2136 April 1997

[9] P. Mockapetris, "*Domain Names – Implementation and specification*", RFC 1035, November 1987

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

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