DHC Working Group
Internet-Draft
Expires: January 2, 2009

D. Rao
B. Joshi
P. Kurapati
Infosys Technologies Ltd.
July 1, 2008

# DHCPv4 bulk lease query
# draft-dtv-dhc-dhcpv4-bulk-leasequery-00.txt

## Status of this Memo



## Copyright Notice



## Abstract

The Dynamic Host Configuration Protocol for IPv4 (DHCPv4) has been extended with a Leasequery capability that allows a client to request information about DHCPv4 bindings. That mechanism is limited to queries for individual bindings. In some situations individual binding queries may not be efficient, or even possible. This document expands on the Leasequery protocol, adding new query types and allowing for bulk transfer of DHCPv4 binding data via TCP.

Table of Contents

1. Introduction . . . . . 3
2. Terminology . . . . . 4
3. Motivation . . . . . 5
4. Design Goals . . . . . 6
4.1. Information Acquisition before Data Starts . . . . . 6
4.2. Lessen Negative Caching . . . . . 6
4.3. Antispoofing in 'Fast Path' . . . . . 6
5. Protocol Overview . . . . . 7
6. Interaction Between UDP Leasequery and Bulk Leasequery . . . . 9
7. Message and Option Definitions . . . . . 10
7.1. Message Framing for TCP . . . . . 10
7.2. Messages . . . . . 11
7.2.1. DHCPLEASEDATA . . . . . 12
7.2.2. DHCPLEASEDONE . . . . . 12
7.2.3. DHCPLEASEQUERYFAIL . . . . . 12
7.3. Query Types . . . . . 13
7.3.1. QUERY_BY_RELAYID . . . . . 13
7.3.2. QUERY_BY_REMOTE_ID . . . . . 14
7.4. Options . . . . . 14
7.4.1. STATUS-CODE option . . . . . 14
7.5. Connection and Transmission Parameters . . . . . 16
8. Requestor Behavior . . . . . 17
8.1. Connecting . . . . . 17
8.2. Forming Queries . . . . . 17
8.3. Processing Replies . . . . . 17
8.4. Leasequery Request Completion Criteria . . . . . 18
8.5. Querying Multiple Servers . . . . . 19
8.6. Multiple Queries to a Single Server . . . . . 19
8.6.1. Example . . . . . 19
8.7. Closing Connections . . . . . 20
9. Server Behavior . . . . . 21
9.1. Accepting Connections . . . . . 21
9.2. Forming Replies . . . . . 21
9.3. Multiple or Parallel Queries . . . . . 22
9.4. Closing Connections . . . . . 22
10. Security Considerations . . . . . 24
11. IANA Considerations . . . . . 25
12. Acknowledgments . . . . . 26
13. References . . . . . 27
13.1. Normative Reference . . . . . 27
13.2. Informative Reference . . . . . 27
Appendix 1. Why a New Leasequery is Required? . . . . . 28
Authors' Addresses . . . . . 31
Intellectual Property and Copyright Statements . . . . . 32

## 1. Introduction

The DHCPv4 [2] protocol specifies a mechanism for the assignment of IPv4 address and configuration information to IPv4 nodes. DHCPv4 servers maintain authoritative binding information.

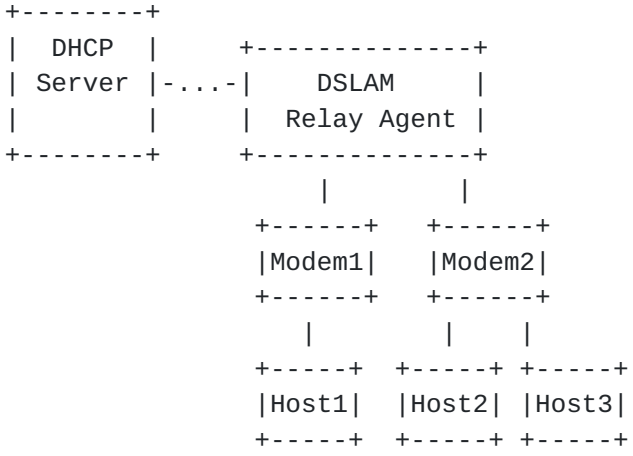



Figure 1

DHCP relay agents snoop DHCP messages and append relay agent information option before relaying it to the configured DHCP Servers (see Figure 1). In this process, some relay agents also glean the lease information sent by the server and maintain this locally. This information is used for prevention of spoofing attempts from the clients and to install routes. When a relay agent reboots, this information is lost. RFC 4388 [5] has defined a mechanism to obtain this lease information from the server. The existing query types in leasequery are data driven; relay agent initiates the leasequery when it receives data traffic from/for the client. This approach does not scale well when there are thousands of clients connected to the relay agent. Different query types are needed where a relay agent can query the server without waiting for the traffic from/for the clients.

This document extends the DHCPv4 Leasequery protocol to add support for queries that address these requirements. There may be many thousands of DHCPv4 bindings per relay agent, so we specify the use of TCP [4] for efficiency of data transfer. We specify a new query type that uses the Relay Identifier sub-option to support efficient recovery of all data associated with a specific relay agent. We also specify query-type by Remote-ID sub-option value, to assist a relay agent that needs to recover a subset of its clients' bindings.

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [1].

DHCPv4 terminology is defined in [2]. DHCPv4 Leasequery terminology is defined in [5]. DUID terminology is in defined in [7]. Relay agent terminalogy is defined in [3].

## 3. Motivation

Consider a typical DSLAM working also as a DHCP relay agent (see Figure 1). "Fast path" and "slow path" generally exist in most networking boxes including DSLAMs. Fast path processing is done in network processor or an ASIC (Application Specific Integrated Circuit). Slow path processing is done in a normal processor. As much as possible, regular data handling code should be in fast path. Slow path processing should be reduced as it may become a bottleneck.

For a DSLAM having multiple DSL ports, multiple IP addresses may be assigned using DHCP to a single port and the number of clients on a port may be unknown. The DSLAM may also not know the network portions of the IP addresses that are assigned to its DHCP clients.

The DSLAM gleans IP address or other information from DHCP negotiations for antispoofing and for other purposes. The antispoofing itself is done in fast path. DSLAM keeps track of only one list of IP addresses: list of IP addresses that are assigned by DHCP server. Traffic for all other IP addresses is dropped. If client starts its data transfer after its DHCP negotiations are gleaned by DSLAM, no legitimate packets will be dropped because of antispoofing. In other words, antispoofing is effective (no legitimate packets are dropped and all spoofed packets are dropped) and efficient (antispoofing is done in fast path). The intention is to achieve similar effective and efficient antispoofing in the lease query scenario also when a DSLAM loses its gleaned information (for example, because of reboot).

After a deep analysis, we found that the three existing query types supported by RFC 4388 [5] do not provide effective and efficient antispoofing for the above scenario and a new mechanism is required.

The existing query types

- necessitate a data driven approach: the lease queries can only be done when the Access Concentrator receives data. That results in increased outage time for clients.

- result in excessive negative caching consuming lot of resources under a spoofing attack.

- result in antispoofing being done in slow path instead of fast path.

The deeper analysis, which led to the above conclusions, itself appears as an Appendix to this document.

## 4. Design Goals

The goal of this document is to provide a lightweight mechanism for an Access Concentrator to retrieve lease information available in the DHCP server. The mechanism should also support an Access Concentrator to retrieve consolidated lease information for the entire access concentrator or for a connection/circuit.

### 4.1. Information Acquisition before Data Starts

Existing data driven approach by RFC 4388 [5] means that the lease queries can only be done when an Access Concentrator receives data. For antispoofing, packets need to be dropped until it gets the lease information from DHCP server. If an Access Concentrator finishes the lease queries before it starts receiving data, then there is no need to drop legitimate packets. So, effectively outage time may be reduced.

### 4.2. Lessen Negative Caching

If lease queries result in negative caches, then that puts additional overhead on the access concentrator. The negative caches not only consume precious resources they also need to be managed. Hence they should be avoided as much as possible. The lease queries should reduce the need for negative caching as far as possible.

### 4.3. Antispoofing in 'Fast Path'

If Antispoofing is not done in fast path, it will become a bottleneck and may lead to denial of service of the access concentrator. The lease queries should make it possible to do antispoofing in fast path.

## 5. Protocol Overview

The Bulk Leasequery mechanism is modeled on the existing individual Leasequery protocol described in RFC 4388[5]; most differences arise from the use of TCP. A Bulk Leasequery client opens a TCP connection to a DHCPv4 Server, using the DHCPv4 port 67. Note that this implies that the Leasequery client has server IP address(es) available via configuration or some other means, and that it has unicast IP reachability to the server. No relaying for bulk leasequery is specified.

After establishing a connection, the client sends a DHCPLEASEQUERY message containing a query-type and data about bindings it is interested in. The server uses the query-type and the data to identify any relevant bindings. In order to support some query-types, servers may have to maintain additional data structures or be able to locate bindings based on specific option data. The server replies with a DHCPLEASEUNASSIGNED, DHCPLEASEACTIVE, DHCPLEASEQUERYFAIL, or DHCPLEASEUNKNOWN. The reasons why a DHCPLEASEUNASSIGNED, DHCPLEASEACTIVE, or DHCPLEASEUNKNOWN message might be generated are explained in [5] and below. The reasons why a DHCPLEASEQUERYFAIL message might be generated are explained below. If the query was successful, the server includes the first client's binding data in a DHCPLEASEACTIVE message. If more than one client's bindings are being returned, the server then transmits the additional client bindings in a series of DHCPLEASEDATA messages. If the server has sent at least one client's bindings, it sends a DHCPLEASEDONE message when it has finished sending its replies. The client may reuse the connection to send additional queries. Each end of the TCP connection can be closed after all data has been sent.

The Relay-ID sub-option is defined in [6]. The sub-option contains a DUID identifying a DHCPv4 relay agent. Relay agents can include this sub-option while relaying messages to DHCPv4 servers. Servers can retain the Relay-ID and associate it with bindings made on behalf of the relay agent's clients. A relay agent can then recover binding information about downstream clients by using the Relay-ID in a DHCPLEASEQUERY message.

Bulk Leasequery supports the queries by IPv4 address, MAC address, and Client Identifier as specified in RFC4388 [5]. The Bulk Leasequery protocol also adds two new queries.

o Query by Relay Identifier

This query asks a server for the bindings associated with a specific relay agent; the relay agent is identified by a DUID carried in a Relay-ID sub-option.

- Query by Remote ID

This query asks a server for the bindings associated with a Relay Agent Remote-ID sub-option [3] value.

## 6. Interaction Between UDP Leasequery and Bulk Leasequery

Bulk Leasequery can be seen as an extension of the existing UDP Leasequery protocol [5]. This section tries to clarify the relationship between the two protocols.

The query-types introduced in the UDP Leasequery protocol can be used in the Bulk Leasequery protocol. One change in behavior is required when Bulk Leasequery is used. RFC4388 [5], in sections 6.1, 6.4.1, and 6.4.2 specifies the use of an associated-ip option in DHCPLEASEACTIVE messages in cases where multiple bindings were found. When Bulk Leasequery is used, this mechanism is not necessary: a server returning multiple bindings simply does so directly as specified in this document. The associated-ip option MUST NOT appear in Bulk Leasequery replies.

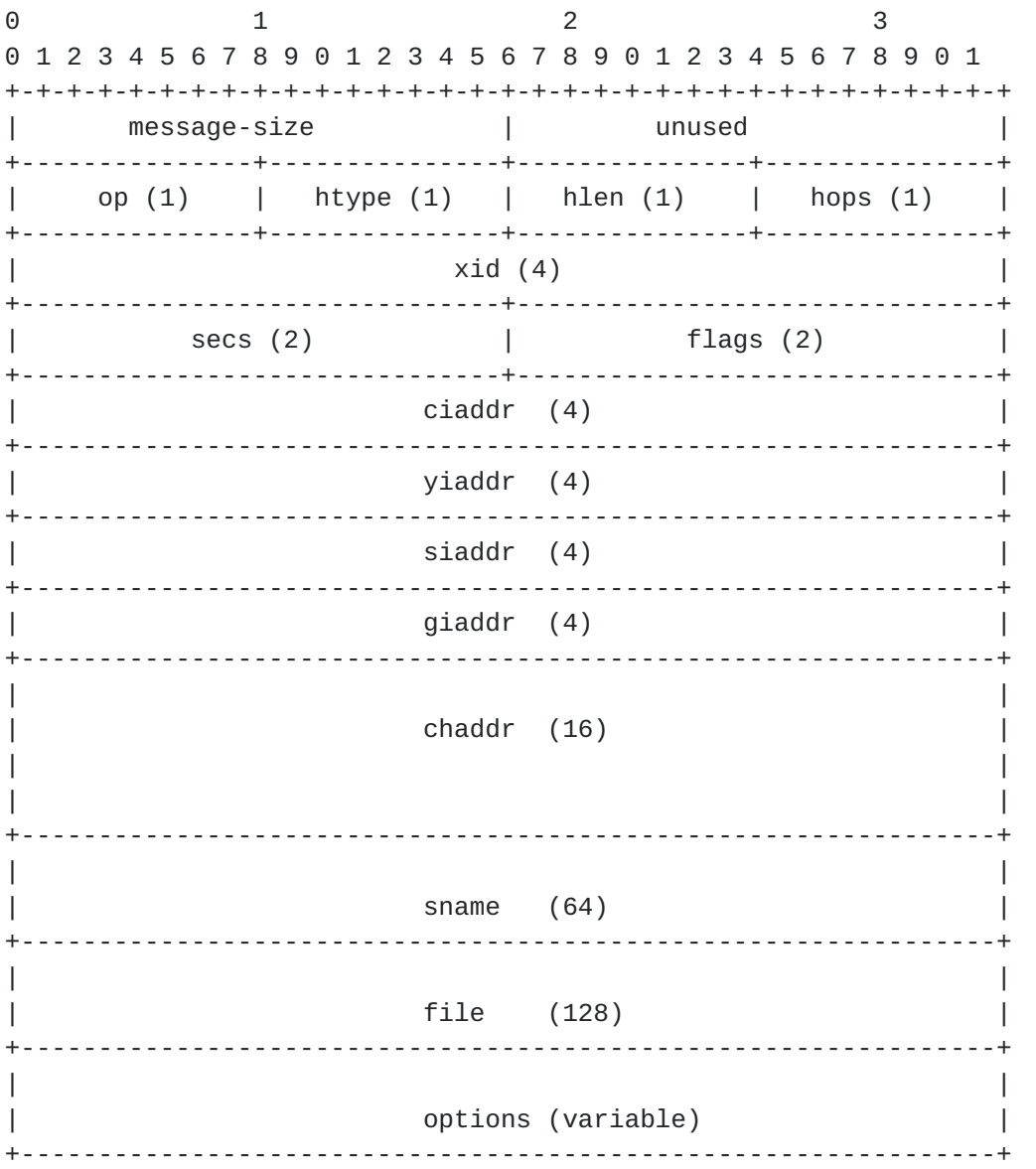
Only DHCPLEASEQUERY, DHCPLEASEACTIVE, DHCPLEASEUNASSIGNED, DHCPLEASEUNKNOWN, DHCPLEASEDATA, DHCPLEASEQUERYFAIL, and DHCPLEASEDONE messages are allowed over the Bulk Leasequery connection. No other DHCPv4 messages are supported. The Bulk Leasequery connection is not an alternative DHCPv4 communication option for clients seeking DHCPv4 service.

# 7. Message and Option Definitions

## 7.1. Message Framing for TCP

The use of TCP for the Bulk Leasequery protocol permits one or more DHCPv4 messages to be sent at a time. The receiver needs to be able to determine how large each message is. Four octets, out of which the first two octets contain the message size in network byte-order, are prepended to each DHCPv4 message sent on a Bulk Leasequery TCP connection. The four octets 'frame' each DHCPv4 message.

DHCPv4 message framed for TCP:




message-size the number of octets in the message that follows (excluding the two unused bytes), as a 16-bit integer in network byte-order.

unused these 16 bits are unused; they should be set to zero by sender and ignored by receiver.

All other fields are as specified in DHCPv4 [2].

## 7.2. Messages

The DHCPLEASEQUERY, DHCPLEASEUNASSIGNED, DHCPLEASEUNKNOWN, and DHCPLEASEACTIVE messages are defined in RFC4388 [5]. In a Bulk Leasequery exchange, a single DHCPLEASEUNASSIGNED, DHCPLEASEQUERYFAIL, or DHCPLEASEUNKNOWN message is used to indicate

the failure of a query. A single DHCPLEASEACTIVE message is used to indicate the success of a query, and contains the first client's binding data. It also carries data that do not change in the context of a single query and answer, such as the Server Identifier (option 54) option.

### 7.2.1. DHCPLEASEDATA

The DHCPLEASEDATA message (message type TBD) carries data about a single DHCPv4 client's leases and bindings. The purpose of the message is to reduce redundant data when there are multiple bindings to be sent. The DHCPLEASEDATA message MUST be preceded by a DHCPLEASEACTIVE message. The DHCPLEASEACTIVE carries the Leasequery's Server Identifier options, and carries the first client's binding data if the query was successful.

DHCPLEASEDATA MUST ONLY be sent in response to a successful DHCPLEASEQUERY, and only if more than one client's data is to be sent. The DHCPLEASEDATA message's xid field MUST match the xid of the DHCPLEASEQUERY request message. The Server Identifier option SHOULD NOT be included: that data should be constant for any one Bulk Leasequery reply, and should have been conveyed in the DHCPLEASEACTIVE message.

### 7.2.2. DHCPLEASEDONE

The DHCPLEASEDONE message (message type TBD) indicates the end of a group of related Leasequery replies. The DHCPLEASEDONE message's xid field MUST match the xid of the DHCPLEASEQUERY request message. The presence of the message itself signals the end of a stream of reply messages. A single DHCPLEASEDONE MUST be sent after all replies (a DHCPLEASEACTIVE and zero or more DHCPLEASEDATA messages) to a successful Bulk Leasequery request that returned at least one binding. Other DHCPv4 options SHOULD NOT be included in the DHCPLEASEDONE message.

### 7.2.3. DHCPLEASEQUERYFAIL

A server may encounter an error condition while processing a DHCPLEASEQUERY message but before it has sent any response. A server may also encounter an error condition after it has sent the initial DHCPLEASEACTIVE. In these cases, it SHOULD attempt to send a DHCPLEASEQUERYFAIL (message type TBD) with STATUS_CODE option indicating the error condition to the requestor. Other DHCPv4 options SHOULD NOT be included in the DHCPLEASEQUERYFAIL message.

### 7.3. Query Types

We introduce the following new query-types: QUERY_BY_RELAYID and QUERY_BY_REMOTE_ID. These queries are designed to assist relay agents in recovering binding data in circumstances where some or all of the relay agent's binding data has been lost.

#### 7.3.1. QUERY_BY_RELAYID

This query asks the server to return bindings associated with the specified relay DUID. A relay agent MAY include the option in the messages it relays. Obviously, it will not be possible for a server to respond to QUERY_BY_RELAYID queries unless the relay agent has included this option. A relay agent SHOULD be able to generate a DUID for this purpose, and capture the result in stable storage. A relay agent SHOULD also allow the DUID value to be configurable: doing so allows an administrator to replace a relay agent while retaining the association between the relay agent and existing DHCPv4 bindings.

A DHCPv4 Server MAY associate Relay-ID sub-options from relayed messages it processes with lease bindings that result. Doing so allows it to respond to QUERY_BY_RELAYID Leasequeries.

QUERY_BY_RELAYID message is formatted as follows:

- The requester supplies only an option 82 which will include only an Agent Relay ID sub-option in the DHCPLEASEQUERY message. The query options MUST contain a RELAYID sub-option.

- The "ciaddr" field MUST be set to zero.

- The values of htype, hlen, and chaddr MUST be set to zero.

- The Client-identifier option (option 61) MUST NOT appear in the packet.

The DHCP server replies with a DHCPLEASEACTIVE message if the DHCP server has one or multiple active leases which were assigned through the Relay Agent identified by Relay ID in the DHCPLEASEQUERY message. If the Server has recorded Relay-ID values with its bindings, it uses the sub-option's value to identify bindings to return. Server replies with a DHCPLEASEUNASSIGNED if it has information of the said relay ID but no lease is associated with the same. Server replies with a DHCPLEASEUNKNOWN message if it has no information of the said relay ID.

### 7.3.2. QUERY_BY_REMOTE_ID

The QUERY_BY_REMOTE_ID is used to ask the server to return bindings associated with a Remote-ID sub-option value from a relayed message. QUERY_BY_REMOTE_ID for TCP defined in this draft is consistent with QUERY_BY_REMOTE_ID for UDP defined in [9].

In order to support this query, a server needs to record the most-recent Remote-ID sub-option value seen in a relayed message along with its other binding data.

QUERY_BY_REMOTE_ID message is formatted as follows:

- There MUST be only a Relay Agent Information option (option 82) with only Agent Remote ID sub-option (sub-option 2) in the DHCPLEASEQUERY message.
- The "ciaddr" field MUST be set to zero.
- The values of htype, hlen, and chaddr MUST be set to zero.
- The Client-identifier option (option 61) MUST NOT appear in the packet.

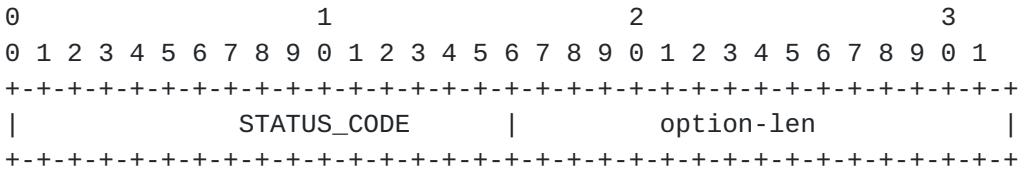
The DHCP server replies with a DHCPLEASEACTIVE message if the Agent Remote ID in the DHCPLEASEQUERY message currently has an active lease on an IP address in this DHCP server. Server replies with a DHCPLEASEUNASSIGNED if it has information of the said remote ID but no lease is associated with the same. Server replies with a DHCPLEASEUNKNOWN message if it has no information of the said remote ID. If the Server has recorded Remote-ID values with its bindings, it uses the sub-option's value to identify bindings to return.

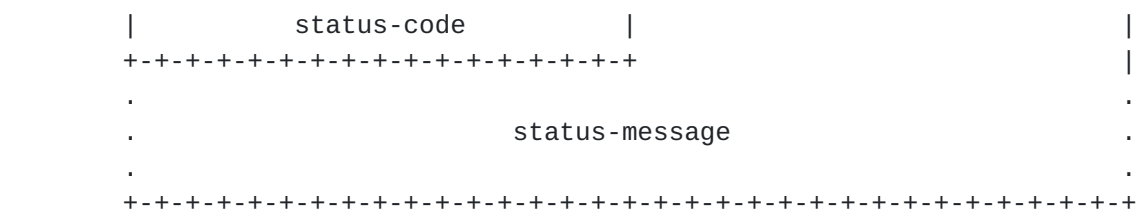
## 7.4. Options

### 7.4.1. STATUS-CODE option

This option returns a status indication related to the DHCP message. Currently it is used along with DHCPLEASEQUERYFAIL message.

The format of the Status Code option is:






option-code STATUS_CODE (TBD).

option-len 2 + length of status-message.

status-code The numeric code for the status encoded in this option. The status codes are defined below.

status-message A UTF-8 encoded text string suitable for display to an end user, which MUST NOT be null-terminated.

A Status Code option may appear in the options field of a DHCP message. If the Status Code option does not appear in a message in which the option could appear, the status of the message is assumed to be Success.

The following status codes are defined:

| Name | Code | Description |
|---|---|---|
| Success | 0 | Success. |
| UnspecFail | 1 | Failure, reason unspecified; this status code is sent by either a client or a server to indicate a failure not explicitly specified in this document. |
| UnknownQueryType | 2 | The query-type is unknown to or not supported by the server. |
| MalformedQuery | 3 | The query is not valid; for example, a required query option is missing. |
| NotAllowed | 4 | The server does not allow the requestor to issue this DHCPLEASEQUERY |
| QueryTerminated | 5 | Indicates that the server is unable to perform a query or has prematurely terminated the query for some reason (which should be communicated in the text message). This may be because the server is short of resources or is being shut down. The requestor may retry the query at a later time. The requestor should wait at least a short interval before retrying. Note that while a server may simply prematurely close |

its end of the connection, it is preferable for the server to send a DHCPLEASEQUERYFAIL with this status-code to notify the requestor of the condition.

## 7.5. Connection and Transmission Parameters

DHCPv4 Servers that support Bulk Leasequery SHOULD listen for incoming TCP connections on the DHCPv4 server port 67. Implementations MAY offer to make the incoming port configurable, but port 67 MUST be the default. Client implementations SHOULD make TCP connections to port 67, and MAY offer to make the destination server port configurable.

This section presents a table of values used to control Bulk Leasequery behavior, including recommended defaults. Implementations MAY make these values configurable.

| Parameter | Default | Description |
|---|---|---|
| BULK_LQ_CONN_TIMEOUT | 30 secs | Bulk Leasequery connection timeout |
| BULK_LQ_DATA_TIMEOUT | 30 secs | Bulk Leasequery data timeout |
| BULK_LQ_MAX_RETRY | 60 secs | Max Bulk Leasequery retry timeout |
| BULK_LQ_MAX_CONNS | 10 | Max Bulk Leasequery TCP connections |

## 8. Requestor Behavior

### 8.1. Connecting

A Requestor attempts to establish a TCP connection to a DHCPv4 Server in order to initiate a Leasequery exchange. The Requestor SHOULD be prepared to abandon the connection attempt after BULK_LQ_CONN_TIMEOUT. If the attempt fails, the Requestor MAY retry. Retries MUST use an exponential backoff timer, increasing the interval between attempts up to BULK_LQ_MAX_RETRY.

### 8.2. Forming Queries

After a connection is established, the Requestor constructs a Leasequery message, as specified in [5]. The query may have any of the defined query-types, and includes the options and data required by the query-type chosen. The Requestor sends the message size, 16 reserved bits (zeroes), and then sends the actual DHCPv4 message, as described in Section 7.1.

If the TCP connection becomes blocked while the Requestor is sending its query, the Requestor SHOULD be prepared to terminate the connection after BULK_LQ_DATA_TIMEOUT. We make this recommendation to allow Requestors to control the period of time they are willing to wait before abandoning a connection, independent of notifications from the TCP implementations they may be using.

### 8.3. Processing Replies

The Requestor attempts to read a DHCPLEASEACTIVE, DHCPLEASEUNKNOWN, DHCPLEASEQUERYFAIL, or DHCPLEASEUNASSIGNED message from the TCP connection. If the stream of replies becomes blocked, the Requestor SHOULD be prepared to terminate the connection after BULK_LQ_DATA_TIMEOUT, and MAY begin retry processing if configured to do so.

The Requestor examines the DHCPLEASEACTIVE, DHCPLEASEUNKNOWN, DHCPLEASEQUERYFAIL, or DHCPLEASEUNASSIGNED message, and determines how to proceed. If the xid in the received message does not match an outstanding DHCPLEASEQUERY message, the client MUST close the TCP connection. Message processing rules for DHCPLEASEACTIVE are specified in DHCPv4 Leasequery [5]. DHCPLEASEUNKNOWN and DHCPLEASEUNASSIGNED replies indicate that the target server had no bindings matching the query. DHCPLEASEQUERYFAIL indicates that the server failed to serve the client. More details will be available in the received STATUS_CODE option.

The Leasequery protocol [5] uses the associated-ip option as an

indicator that multiple bindings were present in response to a single query. For Bulk Leasequery, the associated-ip option is not used, and MUST NOT be present in replies.

A successful DHCPLEASEACTIVE that is returning binding data is created as specified in [5]. If there are additional bindings to be returned, they will be carried in DHCPLEASEDATA messages. Each DHCPLEASEDATA message returns binding data and is prepared just like the DHCPLEASEACTIVE message described in [5] except for the new message type.

A single bulk query can result in a large number of replies. For example, a single relay agent might be responsible for thousands of DHCP clients. The Requestor MUST be prepared to receive more than one DHCPLEASEDATA with xids matching a single DHCPLEASEQUERY message.

The DHCPLEASEDONE message ends a successful Bulk Leasequery request that returned at least one binding. DHCPLEASEUNKNOWN and DHCPLEASEUNASSIGNED MUST NOT be followed by a DHCPLEASEDONE message for the same xid. After receiving DHCPLEASEDONE from a server, the Requestor MAY close the TCP connection to that server. If the xid in the DHCPLEASEDONE does not match an outstanding DHCPLEASEQUERY message, the client MUST close the TCP connection.

The DHCPLEASEQUERYFAIL message ends a Bulk Leasequery request in failure. Depending on the status code, the requestor may try a different server (such as for NotAllowed and UnknownQueryType), try a different or corrected query (such as for UnknownQueryType and MalformedQuery), or terminate the query. If the xid in the DHCPLEASEQUERYFAIL does not match an outstanding DHCPLEASEQUERY message, the client MUST close the TCP connection.

### 8.4. Leasequery Request Completion Criteria

This section provides rules for when a DHCPLEASEQUERY request is complete.

A DHCPLEASEQUERY request is complete for a requestor (i.e., no further messages for that request will be received):

- If it receives a DHCPLEASEUNASSIGNED, DHCPLEASEUNKNOWN, DHCPLEASEDONE, or DHCPLEASEQUERYFAIL message.

- If the connection is closed.

### 8.5. Querying Multiple Servers

A Bulk Leasequery client MAY be configured to attempt to connect to and query from multiple DHCPv4 servers in parallel. Binding data received from multiple DHCPv4 servers may need to be reconciled.

The client data from the different servers may be disjoint or overlapping.

When using the DHCPLEASEQUERY message in an environment where multiple DHCP servers may contain authoritative information about the same IP address (such as when two DHCP servers are cooperating to provide a high-availability DHCP service), multiple, possibly conflicting, responses might be received.

In this case, some information in the response packet SHOULD be used to decide among the various responses. The client-last-transaction-time (if it is available) can be used to decide which server has more recent information concerning the IP address returned in the "ciaddr" field.

If the requestor receives disjoint client data from different sources, it SHOULD merge them.

### 8.6. Multiple Queries to a Single Server

Bulk Leasequery clients may need to make multiple queries in order to recover binding information. A Requestor MAY use a single connection to issue multiple queries. Each query MUST have a unique xid. A server MAY process more than one query at a time. A server that is willing to do so MAY interleave replies to the multiple queries within the stream of reply messages it sends. Clients need to be aware that replies for multiple queries may be interleaved within the stream of reply messages. Clients that are not able to process interleaved replies (based on xid) MUST NOT send more than one query at a time. Requestors should be aware that servers are not required to process queries in parallel, and that servers are likely to limit the rate at which they process queries from any one Requestor.

#### 8.6.1. Example

This example illustrates what a series of queries and responses might look like. This is only an example - there is no requirement that this sequence must be followed, or that clients or servers must support parallel queries.

In the example session, the client sends four queries after establishing a connection. Query 1 results in a failure; query 2

succeeds and the stream of replies concludes before the client issues any new query. Query 3 and query 4 overlap, and the server interleaves its replies to those two queries.

```
Client                        Server
------                        ------
DHCPLEASEQUERY xid 1 ----->
                     <-----       DHCPLEASEUNKNOWN xid 1
DHCPLEASEQUERY xid 2 ----->
                     <-----       DHCPLEASEACTIVE xid 2
                     <-----       DHCPLEASEDATA xid 2
                     <-----       DHCPLEASEDATA xid 2
                     <-----       DHCPLEASEDONE xid 2
DHCPLEASEQUERY xid 3 ----->
DHCPLEASEQUERY xid 4 ----->
                     <-----       DHCPLEASEACTIVE xid 4
                     <-----       DHCPLEASEDATA xid 4
                     <-----       DHCPLEASEACTIVE xid 3
                     <-----       DHCPLEASEDATA xid 4
                     <-----       DHCPLEASEDATA xid 3
                     <-----       DHCPLEASEDONE xid 3
                     <-----       DHCPLEASEDATA xid 4
                     <-----       DHCPLEASEDONE xid 4
```

### 8.7. Closing Connections

The Requestor MAY close its end of the TCP connection after sending a DHCPLEASEQUERY message to the server. The Requestor MAY choose to retain the connection if it intends to issue additional queries. Note that this client behavior does not guarantee that the connection will be available for additional queries: the server might decide to close the connection based on its own configuration.

## 9. Server Behavior

### 9.1. Accepting Connections

Servers that implement DHCPv4 Bulk Leasequery listen for incoming TCP connections. Port numbers are discussed in Section 7.5. Servers MUST be able to limit the number of currently accepted and active connections. The value BULK_LQ_MAX_CONNS MUST be the default; implementations MAY permit the value to be configurable.

Servers MAY restrict Bulk Leasequery connections and DHCPLEASEQUERY messages to certain clients. Connections not from permitted clients SHOULD be closed immediately, to avoid server connection resource exhaustion. Servers MAY restrict some clients to certain query types. Servers MAY reply to queries that are not permitted with the DHCPLEASEQUERYFAIL message with the NotAllowed status code, or MAY close the connection.

If the TCP connection becomes blocked while the server is accepting a connection or reading a query, it SHOULD be prepared to terminate the connection after BULK_LQ_DATA_TIMEOUT. We make this recommendation to allow Servers to control the period of time they are willing to wait before abandoning an inactive connection, independent of the TCP implementations they may be using.

### 9.2. Forming Replies

The DHCPv4 Leasequery [5] specification describes the initial construction of DHCPLEASEACTIVE, DHCPLEASEUNKNOWN, and DHCPLEASEUNASSIGNED messages and the processing of QUERY_BY_IP_ADDRESS, QUERY_BY_MAC_ADDRESS, and QUERY_BY_CLIENTIDENTIFIER. Use of the DHCPLEASEACTIVE and DHCPLEASEDATA messages to carry multiple bindings are described in Section 7.2. Message transmission and framing for TCP is described in Section 7.1. If the connection becomes blocked while the server is attempting to send reply messages, the server SHOULD be prepared to terminate the TCP connection after BULK_LQ_DATA_TIMEOUT.

If the server encounters an error during initial query processing, before any reply has been sent, it SHOULD send a DHCPLEASEQUERYFAIL containing an appropriate STATUS_CODE option. This signals to the requestor that no data will be returned. If the server encounters an error while processing a query that has already resulted in one or more reply messages, the server SHOULD send a DHCPLEASEQUERYFAIL containing an appropriate STATUS_CODE option. The server SHOULD close its end of the connection as an indication that it was not able to complete query processing.

If the server does not recognize the identifier (relay id or remote id) in a query, it SHOULD send a DHCPLEASEUNKNOWN. If the server recognizes the identifier in a query but does not find any bindings satisfying a query, it SHOULD send a DHCPLEASEUNASSIGNED. Otherwise, the server sends each binding's data in a reply message. The first reply message is a DHCPLEASEACTIVE. The binding data is carried as specified in [5] and extended below. The server returns subsequent bindings in DHCPLEASEDATA messages.

For QUERY_BY_RELAYID, the server locates each binding associated with the query's Relay-ID sub-option value. In order to give a meaningful reply to a QUERY_BY_RELAYID, the server has to be able to maintain this association in its DHCPv4 binding data.

For QUERY_BY_REMOTE_ID, the server locates each binding associated with the query's Relay Remote-ID sub-option value. In order to be able to give meaningful replies to this query, the server has to be able to maintain this association in its binding database.

The server sends the DHCPLEASEDONE message as specified in Section 7.2.

### 9.3. Multiple or Parallel Queries

As discussed in Section 6.5, Requestors may want to leverage an existing connection if they need to make multiple queries. Servers MAY support reading and processing multiple queries from a single connection. A server MUST NOT read more query messages from a connection than it is prepared to process simultaneously.

This MAY be a feature that is administratively controlled. Servers that are able to process queries in parallel SHOULD offer configuration that limits the number of simultaneous queries permitted from any one Requestor, in order to control resource use if there are multiple Requestors seeking service.

### 9.4. Closing Connections

The server MAY close its end of the TCP connection after sending its last message (a DHCPLEASEUNASSIGNED, a DHCPLEASEUNKNOWN, DHCPLEASEQUERYFAIL, or a DHCPLEASEDONE) in response to a query. Alternatively, the server MAY retain the connection and wait for additional queries from the client. The server SHOULD be prepared to limit the number of connections it maintains, and SHOULD be prepared to close idle connections to enforce the limit.

The server MUST close its end of the TCP connection if it encounters an error sending data on the connection. The server MUST close its

end of the TCP connection if it finds that it has to abort an in-process request. A server aborting an in-process request MAY attempt to signal that to its clients by using the DHCPLEASEQUERYFAIL message type (Section 7.2.3). If the server detects that the client end has been closed, the server MUST close its end of the connection after it has finished processing any outstanding requests from the client.

## 10. Security Considerations

The "Security Considerations" section of [2] details the general threats to DHCPv4. The DHCPv4 Leasequery specification [5] describes recommendations for the Leasequery protocol, especially with regard to DHCPLEASEQUERY messages, mitigation of packet-flooding DOS attacks, and restriction to trusted clients.

The use of TCP introduces some additional concerns. Attacks that attempt to exhaust the DHCPv4 server's available TCP connection resources, such as SYN flooding attacks, can compromise the ability of legitimate clients to receive service. Malicious clients who succeed in establishing connections, but who then send invalid queries, partial queries, or no queries at all also can exhaust a server's pool of available connections. We recommend that servers offer configuration to limit the sources of incoming connections, that they limit the number of accepted connections and the number of in-process queries from any one connection, and that they limit the period of time during which an idle connection will be left open.

## 11. IANA Considerations

IANA is requested to assign a new DHCPv4 Option Code in the registry maintained in http://www.iana.org/assignments/bootp-dhcp-parameters:

o STATUS_CODE

IANA is requested to assign the following values for the STATUS_CODE option maintained in http://www.iana.org/assignments/bootp-dhcp-parameters:

| | |
|---|---|
| Success | 0 |
| UnspecFail | 1 |
| UnknownQueryType | 2 |
| MalformedQuery | 3 |
| NotAllowed | 4 |
| QueryTerminated | 5 |

IANA is requested to assign values for the following new DHCPv4 Message types in the Message Type option (option 53) in the registry maintained in http://www.iana.org/assignments/bootp-dhcp-parameters:

o DHCPLEASEDONE
o DHCPLEASEDATA
o DHCPLEASEQUERYFAIL

## 12. Acknowledgments

The bulk lease query protocol for DHCPv4 described in this draft is inspired by the bulk lease query protocol defined for DHCPv6 [8] by Mark Stapp and liberally borrows from that draft. We also use the protocol mechanisms proposed in RFC 4388 [5] by R. Woundy and K. Kinnear.

## 13. References

### 13.1. Normative Reference

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

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

[3] Patrick, M., "DHCP Relay Agent Information Option", RFC 3046, January 2001.

[4] Duke, M., Braden, R., Eddy, W., and E. Blanton, "A Roadmap for Transmission Control Protocol (TCP) Specification Documents", RFC 4614, September 2006.

[5] Woundy, R. and K. Kinnear, "Dynamic Host Configuration Protocol (DHCP) Leasequery", RFC 4388, February 2006.

[6] Stapp, M., "The DHCPv4 Relay Agent Identifier Suboption", IETF draft, June 2008.

[7] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 31315, July 2003.

### 13.2. Informative Reference

[8] Stapp, M., "DHCPv6 Bulk Leasequery", IETF draft, June 2008.

[9] Kurapati, P., Desetti, R., and B. Joshi, "DHCPv4 Leasequery by relay agent remote ID", IETF draft, June 2008.

## 1. Why a New Leasequery is Required?

The three existing query types supported by RFC 4388 [5] do not provide effective and efficient antispoofing for the above scenario.

o Query by Client Identifier

Query by Client Identifier is not possible because to use that DSLAM need to glean client identifier also but the whole issue is that we need leasequeries because the gleaned information was lost. On the other hand, we can query by client identifier when client sends a DHCP request, but then there may not be any need for lease query as such -- regular gleaning may be enough.

o Query by IP Address

RFC 4388 [5] suggests that it is preferable to use Query by IP Address when getting downstream traffic.

Query by IP address is not very useful in downstream traffic because downstream traffic may not exist for the clients on a DSL port. (In most Internet applications, downstream traffic exists only when a client sends upstream traffic). In other words, the client will be denied service until it gets downstream traffic, which may never come.

Query by IP address may be used for upstream traffic. Then whenever an upstream packet comes whose IP address is unknown to the DSLAM, a lease query may be initiated. A related question is what to do with that upstream traffic itself until lease query response comes? If the traffic is dropped, we may be dropping legitimate traffic. If the traffic is forwarded, we may be forwarding spoofed packets. Once the lease response comes, subsequent traffic is handled depending on the response. If a DHCPLEASEACTIVE response comes, DSLAM will accept the traffic. If a DHCPLEASEUNASSIGNED response comes, DSLAM will drop the traffic corresponding to the IP address. If a DHCPLEASEUNKNOWN response comes DSLAM may drop the traffic corresponding to the IP address but will have to periodically send the lease query for that IP address again (additional overhead). The process is triggered whenever an unknown IP address comes.

Note that DSLAM needs to keep track of 4 lists of IP addresses: (1) List of IP addresses for which it got DHCPLEASEACTIVE responses; (2) List of IP addresses for which it got DHCPLEASEUNASSIGNED responses; (3) List of IP addresses for which it got DHCPLEASEUNKNOWN responses; (4) All other IP addresses.

This approach may be acceptable if only legitimate traffic is

received. Consider the case when someone sends packets that uses spoofed IP addresses. In that case, lease response will be DHCPLEASEUNASSIGNED or DHCPLEASEUNKNOWN. RFC 4388 [5] suggests usage of negative caching in this regard (which involves additional resources).

In a spoofing type of attack, negative caching information may grow considerably if attacker varies the source IP address. For each such new source IP address, traffic will come to slow path, a new lease query needs to be initiated, response will be processed, and negative caching to be done. That will mean using many resources for negative caching.

RFC 4388 [5] suggests that if the DSLAM knows the network portion of the IP addresses that are assigned to its clients, then some amount of antispoofing can be done in fast path and some lease queries may be avoided. But as indicated before, that information may not always be available to DSLAMs.

Effectively, antispoofing support involves considerable slow path processing and considerable resources tied for negative caching.

RFC 4388 [5] says that DHCP server should be protected from being flooded with too many leasequery requests and DSLAM also should not send too many lease query messages at a time. This would mean that legitimate clients may be excessively delayed getting their information in the face of antispoofing attacks.

It is concluded that antispoofing is neither effective nor efficient with this query type.

o Query by MAC Address

Query by MAC address can also be used similar to query by IP address described above. Indeed, query by MAC address may be better than query by IP address in one sense because of the possible presence of associated-ip option in lease responses (Note that associated-ip option does not appear in responses for query by IP address). With associated-ip option DSLAM can get information not only about the IP address/MAC address that triggered the lease query but also about other IP addresses that are associated with the original MAC address. That way, when traffic that uses the other IP addresses comes along, DSLAM is already prepared to deal with them.

Although, query by MAC address is better than query by IP address in the above respect, it has a specific problem which is not shared by query by IP address. For a query by MAC address, only two types of responses are possible: DHCPLEASEUNKNOWN and DHCPLEASEACTIVE;

DHCPLEASEUNASSIGNED is not supported. This is particularly troublesome when a DHCP server indeed has definitive information that no IP addresses are associated with the specified MAC address in the leasequery, but it is forced to respond with DHCPLEASEUNKNOWN instead of DHCPLEASEUNASSIGNED. As we have seen above, unlike DHCPLEASEUNASSIGNED, DHCPLEASEUNKNOWN requires periodic querying with DHCP server, an additional overhead.

Moreover, query by MAC address also shares all other issues we discussed above for query by IP address.

We conclude that existing lease query types are not appropriate to achieve effective and efficient antispoofing.

Authors' Addresses

Ramakrishna Rao D. T. V.
Infosys Technologies Ltd.
44 Electronics City, Hosur Road
Bangalore 560 100
India

Email: RAMAKRISHNADTV@infosys.com
URI: http://www.infosys.com/

Bharat joshi
Infosys Technologies Ltd.
44 Electronics City, Hosur Road
Bangalore 560 100
India

Email: bharat_joshi@infosys.com
URI: http://www.infosys.com/

Pavan Kurapati
Infosys Technologies Ltd.
44 Electronics City, Hosur Road
Bangalore 560 100
India

Email: pavan_kurapati@infosys.com
URI: http://www.infosys.com/

## Intellectual Property Statement



## Disclaimer of Validity



## Copyright Statement



## Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.