Internet Draft   Bernie Volz
                                                     Steve Gonczi
                                                 Process Software

                                                        Ted Lemon
                                           Internet Engines, Inc.

                                                      Rob Stevens
                                               Join Systems, Inc.

March 2000   Expires Sept 2000

DHC load balancing algorithm
<draft-ietf-dhc-loadb-01.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."
The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt
The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Copyright Notice
Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

This draft proposes a method of algorithmic load balancing.
It enables multiple, cooperating servers to decide which one should service a client, without exchanging any information beyond initial configuration.

The draft proposes a computable server selection mechanism when multiple DHCP servers are available to service DHCP clients. In addition, it offers the same mechanism select the target server of a forwarding agent such as a BOOTP relay. The possible benefits overlap with those enumerated in [SSO-03], but this draft does not require any DHCP client modifications.

1. Introduction

This protocol was originally devised to support a specific load balancing optimization of the DHC Failover Protocol [FAILOVR]. The authors later realized that it could be used to optimize the behavior of cooperating DHCP servers and the BOOTP relay agents that forward packets to them. The proposal makes it possible to set up each participating server to accept a pre-configured (approximate) percentage of the client load. This is done using a deterministic hashing algorithm, The algorithm could easily be applied to other protocols, having similar characteristics.

2. Terminology

This section discusses both the generic requirements terminology common to many IETF protocol specifications, and also terminology introduced by this document.

2.1. Requirements 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 RFC 2119 [RFC 2119].

2.2. Load balancing terminology

This document introduces the following terms:

Service Delay, SD
   A load balancing parameter, allowing delayed service of a client by a server participating in the load balancing scheme, instead of ignoring the client.

Hash Bucket Assignments, HBA
   A configuration directive that assigns a set of hash bucket values to a server participating in the load balancing scheme.

Server ID, SID
   An identifier that can be used to designate one of the participating Servers. In the context of DHCP, the SID is the IP address or DNS name of the server.

Service Transaction, ST
   A set of client-server exchanges that lead to a server providing or denying some service to a client. Example: the DISCOVER/OFFER/ REQUEST/ACK message exchange between a DHCP server and client is a service transaction.

Service Transaction ID, STID
An attribute of the individual client requests used for load balancing.

3. Background and External Requirements

Because DHCP clients use UDP broadcast to contact DHCP servers, a client DHCPDISCOVER message may be received by more than one server. All servers receiving such a broadcast may respond to the client, letting the client chooses which server it will use.

When a BOOTP relay agent is used, it typically forwards or rebroadcasts client broadcasts to all configured servers, so a similar inefficiency is present.

The optimization described allows a server to be chosen for each such transaction by performing a "serve" / "do not serve" computation. A forwarding agent can perform the same computation to choose a forwarding destination.

In either case, the choice of server can be computed, without the participants having to negotiate who is to respond.

The approach is probabilistic in nature, because it is nearly impossible to foresee which client will request service next. For short periods of time, the actual percentage of clients served by a given server will likely deviate from the desired percentage. As the number of requests grows, the actual percentage of the load being handled by each server will approximate the configured percentage.

4. Overview

DHCP servers MUST use the Client Identifier option as the STID if it is present. If no Client Identifier option is present, the hlen field of the DHCP packet MUST be used as the length of the data to be hashed, and the contents of the chaddr MUST be the data to be hashed. The number of bytes hashed MUST NOT exceed sixteen.

The proposal maps the STID into a hash value using the function in section 6. The resulting hash value can then be used to decide who should respond to the request, or who the forwarding target should be.

The provided hash function generates hash values 0 to 255, and yields a fairly even hash bucket distribution for random STID-s, and also for STID sequences that have some pattern. Resource allocation is accomplished by assigning a set of specific hash values to each participating server.

A server will only service a requests if the STID hash of the request does match one its assigned hash values.

Any hash buckets not assigned to servers will result in some client ST-s being entirely ignored. (In some scenarios, this may be a desirable outcome). STID-s need not be unique, but should have sufficient variety to distribute load to each server.

HBA-s MAY be transmitted as messages, encapsulated in messages of another protocol, e.g.: e-mail, or DHC Failover Protocol option.

DHCP server implementations may optionally be configurable to handle a case where load balancing is being done but the server that is supposed to respond is not available, or is out of suitable addresses.

DHCP server implementations that provide this capability SHOULD set the DS (Delayed Service) configuration parameter to the number of seconds to wait after the client's first request has been sent before responding to a client, whose hash would not normally permit the client to be served.

A DHCP server providing this capability SHOULD use the value in the secs field of the client request if its value is not zero. Because some clients may not correctly implement the secs field, a DHCP server MAY keep track of the first instance of a client transaction to which it would not normally respond. If the server receives a request from a client that has the same transaction ID as a previously recorded request, and if the secs field in the second packet is zero, the DHCP server MAY use the time in seconds between receipt of the first client request and the receipt of the subsequent client request in place of the secs field in order to determine whether or not to respond.

5. Operation

5.1 Configuration

The configuration step consists of assigning hash values to available servers. This is accomplished by providing one or more Hash Bucket Assignments (HBA-s). (These may come from a configuration file, the Windows NT registry, EEPROM, etc.)

5.2 HBA intended for a server

When configuring one specific server, an HBA in the form of a simple bit map of 32 octet values SHOULD be used.

The first octet in the HBA bitmap represents HBA values 0-7, the next byte values 8-15, and so on, with the thirty-second octet representing values 248-255. In each octet, the least significant bit in that octet represents the largest HBA value in that octet.

Each bit of the HBA is associated with one possible hash value. If a bit is set in the map, it means the recipient server MUST service each client request, where the STID yields the corresponding hash value.

For example, if a server receives a HBA with the following 32 octets:

```
                                                     buckets
       FF FF FF FF FF FF 00 00          ( 0   - 63 )
       FF FF FF FF FF FF FF FF          ( 64  - 127 )
       00 00 00 00 00 00 00 00          ( 128 - 191 )
       00 00 00 00 00 00 00 00          ( 192 - 255 )
```

then it MUST service any client requests where the STID hashes into the bucket values of 0 through 47 and **64 through 127.**

The format of the option SHOULD be as follows:

```
    Code        Len        Hash Buckets
   +-----+-----+-----+-----+----+-----+-----+-----+
   |  0  |  10 |  0  |  32 | b1 |  b2 | ... | b32 |
   +-----+-----+-----+-----+----+-----+-----+-----+
```

The option code and length are 2 byte NBO values. The option number is assigned in the option number space of [FAILOVR].

5.3 Delayed Service parameter

The Delayed Service parameter is optional. If it is not sent, the HBA sets up a strict Server/ Do not serve policy.

If the parameter is used, it MUST be sent immediately before the HBA. The server, who is not supposed to serve a specific request (based on the HBA, and the ST hash), is allowed to respond, after S seconds have elapsed since the client first attempted to get service.
A server MAY use the secs field in the BOOTP header for determining the time since the client has been trying to get service, or it MAY track repeated requests some other way.

Format:

```
     Code       Len         Seconds
   +-----+-----+-----+-----+----+
   |  0  |  30 |  0  |  1  | S  |
   +-----+-----+-----+-----+----+
```

The option code and length are 2 byte NBO values.
The option number is assigned in the option number
space of [FAILOVR].

S is a one byte value, 1..255. It represents the number of
seconds to delay service. The server MAY serve a client after
S seconds elapsed from the client's first request.

5.4 HBA intended for a forwarder

When configuring a forwarding agent, (e.g.: BOOTP relay)
HBA-s consisting of pairs of Server-ID / Hash Bucket values
MAY be used.

Here, the Server ID (SID) designates the server responsible for
the specified Hash Bucket. The forwarding agent
forwards each client request, where the STID yields the
specified hash value, to the server designated by the SID.

The Server ID may be any unique server attribute,
(E.g.: IP address, DNS name, etc) that is meaningful in the context of
the relay agent operation.

A forwarder may be configured to forward a packet to
more than one server. For example, a BOOTP relay could be
set up to split the load between 2 primary-backup server pairs,
running the DHC Failover Protocol [FAILOVR].

A possible configuration file for a forwarding agent
(e.g.: BOOTP relay) may look like this:

```
192.33.43.11  0  .. 24;
192.33.43.12  25 .. 55;
192.33.43.13  56 ..128;
192.33.43.14  129..255;
```

The above configuration consists of 4 HBA-s. The first HBA states:
"Any Client request, where the STID yields a hash value
0 to 24, will be forwarded to server 192.33.43.11".

6. Hash function for load balancing

The following hash function is a C language implementation of the algorithm known as "Pearson's hash". The Pearson's hash algorithm was originally published in [PEARSON]. To make this proposal work, all interoperable implementations MUST use the same hash function.

```
/* A "mixing table" of 256 distinct values, in pseudo-random order. */
    unsigned char loadb_mx_tbl[256] =
    {
    251, 175, 119, 215,  81,  14,  79, 191, 103,  49,
    181, 143, 186, 157,   0, 232,  31,  32,  55,  60,
    152,  58,  17, 237, 174,  70, 160, 144, 220,  90,
    57,  223,  59,   3,  18, 140, 111, 166, 203, 196,
    134, 243, 124,  95, 222, 179, 197,  65, 180,  48,
     36,  15, 107,  46, 233, 130, 165,  30, 123, 161,
    209,  23,  97,  16,  40,  91, 219,  61, 100,  10,
    210, 109, 250, 127,  22, 138,  29, 108, 244,  67,
    207,   9, 178, 204,  74,  98, 126, 249, 167, 116,
    34,   77, 193, 200, 121,   5,  20, 113,  71,  35,
    128,  13, 182,  94,  25, 226, 227, 199,  75,  27,
     41, 245, 230, 224,  43, 225, 177,  26, 155, 150,
    212, 142, 218, 115, 241,  73,  88, 105,  39, 114,
     62, 255, 192, 201, 145, 214, 168, 158, 221, 148,
    154, 122,  12,  84,  82, 163,  44, 139, 228, 236,
    205, 242, 217,  11, 187, 146, 159,  64,  86, 239,
    195,  42, 106, 198, 118, 112, 184, 172,  87,   2,
    173, 117, 176, 229, 247, 253, 137, 185,  99, 164,
    102, 147,  45,  66, 231,  52, 141, 211, 194, 206,
    246, 238,  56, 110,  78, 248,  63, 240, 189,  93,
     92,  51,  53, 183,  19, 171,  72,  50,  33, 104,
    101,  69,   8, 252,  83, 120,  76, 135,  85,  54,
    202, 125, 188, 213,  96, 235, 136, 208, 162, 129,
    190, 132, 156,  38,  47,   1,   7, 254,  24,   4,
    216, 131,  89,  21,  28, 133,  37, 153, 149,  80,
    170,  68,   6, 169, 234, 151 };


unsigned char loadb_p_hash(unsigned char *key,/* The key to be hashed */
                           int len)           /* Key length in bytes  */
    {
        unsigned char hash  = len;
        int i;
        for( i=len ; i > 0 ;  )
            hash = loadb_mx_tbl  [ hash ^ key[ --i ] ];
        return( hash );
    }
```

7. Security

This proposal in and by itself provides no security, nor does it impact existing security. Servers using this algorithm are responsible for ensuring that if the contents of the HBA are transmitted over the network as part of the process of configuring any server, that message be secured against tampering, since tampering with the HBA could result in denial of service for some or all clients.

8. References

[FAILOVR] Kinnear, K,, Droms, R., Rabil, G., Dooley, M., Kapur, A., Gonczi, S., Volz, B., "DHCP Failover Protocol", Internet Draft <draft-ietf-dhc-failover-06.txt>, March 2000.

[PEARSON] The Communications of the ACM Vol.33, No. 6 (June 1990), pp. 677-680.

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

[RFC2219] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," RFC-2219, March 1997.

[SSO-03] Stump, G., Gupta, P., Droms, R. Sommerfeld, R. "The Server Selection Option for DHCP" <draft-ietf-dhc-sso-03.txt>

9. Acknowledgements

Special thanks to Peter K. Pearson, the author of Pearson's hash who has kindly granted his permission to use this algorithm, free of any encumbrances.

This proposal stems from the original idea of hashing MAC addresses to a single bit by Ted Lemon, during a Failover Protocol discussion held at CISCO Systems in February, 1999. Rob Stevens suggested the potential use of this algorithm for purposes beyond those of the Failover Protocol.

Many thanks to Ralph Droms, Kim Kinnear, Mark Stapp, Glenn Waters, Greg Rabil and Jack Wong for their comments during the ongoing discussions.

10. Full Copyright Statement

Copyright (C) The Internet Society (1999). All Rights Reserved.
This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.
The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.
This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

11. Author's information

Bernie Volz
Steve Gonczi
Process Software Corporation
959 Concord St. Framingham, MA 01701
Phone: (508) 879-6994
EMail: volz@process.com
       gonczi@process.com

Ted Lemon
Internet Engines, Inc.
950 Charter Street Redwood City, CA 94063
Phone: (650) 779 6031
EMail: mellon@isc.org

Rob Stevens
Join Systems, Inc.
1032 Elwell Ct Ste 243 Palo Alto CA 94203
Phone: (650)-968-4470
EMail: robs@join.com