INTERNET-DRAFT
<draft-vinod-carp-v1-03.txt>

Vinod Valloppillil
Microsoft Corporation
Keith W. Ross
University of Pennsylvania

26 Feb 1998
Expires August 1998

Cache Array Routing Protocol v1.0

Status of this Memo

Abstract

This draft documents the Cache Array Routing Protocol (CARP) v1.0 for dividing URL-space among an array of loosely coupled proxy servers.

An HTTP client agent (either a proxy server or a client browser) which implements CARP v1.0 can allocate and intelligently route requests for the correct URLs to any member of the Proxy Array. Due to the resulting sorting of requests through these proxies, duplication of cache contents is eliminated and global cache hit rates are improved.

Table of Contents

1. Overview

The Cache Array Routing Protocol describes a distributed caching protocol based on

1) a known membership list of loosely coupled proxies and
2) a hash function for dividing URL space among those proxies

The Proxy Array Membership Table is defined as a plain ASCII text file retrieved from an Array Configuration URL. This document does NOT describe how this table is constructed, merely the format of the fields used by agents implementing.

The hash function plus routing algorithm defined in this document take member proxies described in the Proxy Array Membership Table and make an on-the-fly determination as to which Proxy Array member should be the proper receptacle for a cached version of a resource keyed by URL.

Downstream agents may then access the cached resource by forwarding the proxied HTTP request [5] for that resource to the appropriate member of the Proxy Array.

2. Proxy Array Membership Table

The Proxy Array Membership Table is a plain-text ASCII file which can be published from a URL.

The format of the table is:

```
Proxy Array Information/<Version number>
ArrayEnabled:  <0 | 1>
ConfigID:  <opaque string>
ArrayName:  <opaque string>
ListTTL:  <minutes until next check>

<name> <IP addr> <listening port> <table URL> <agent str>
<statetime> <status UP | DOWN> <load factor> <cache size>
```

2.1 Global Information

These are fields that describe the array itself and are not specific to any one member of an array

Global information is terminated in the Proxy Array Membership Table by a CR/LF/CR/LF.

2.1.1 Version number

The version number for implementations of this specification is 1.0

2.1.2 ArrayEnabled

This field allows proxies to advertise their implementation of CARP v1 even if they are not members of a Proxy Array.

2.1.3 ConfigID

ConfigID is an opaque number no larger than 32bits similar to an ETag in HTTP 1.1. It is used to track the current state of an Array table and may be used to match multiple yet independently published copies of the Proxy Array Membership Table.

2.1.4 ArrayName

ArrayName is an opaque string which is used to provide a convenient administrative name for a given array.

2.1.5 ListTTL

ListTTL is the number of seconds for which an HTTP client entity should consider the current table image valid. After ListTTL

has expired, that client should retrieve a new copy of the Proxy Array Membership Table.

2.2 Member Information

The following fields are published per member in an array and are separated by single spaces. The end of an array member's record is terminated by a CR/LF.

2.2.1 Name

The name of the proxy server. Typically this is the fully qualified DNS name. Downstream HTTP agents should use resolution of this name to determine how to connect to this proxy.

2.2.2 IP Addr

The IP address that other proxy servers within this array should use to connect to this proxy server. This is necessary for proxy servers which may be hosted on multi-hommed servers where requests are only accepted by one of the interfaces.

If this field is not published in the table, name resolution may be used to find a proxy IP address

2.2.3 Listening Port

The TCP port number this proxy is expecting requests on.

2.2.4 Table URL

A URL which may be maintained by this proxy server on which a copy of the array membership table can be found.

This entry does not have to be unique per proxy server. In such cases, the URL should be identical to the URL from which the downstream client requested this table.

2.2.5 Agent String

An opaque string identifying the vendor / version of the proxy Server in the Array Membership Table.

2.2.7 Statetime

How long a Proxy Server has been in its current state and has been a member of this table. This is useful for dynamic generation of the Array Membership Table where the host generating the table has knowledge of the proxy's operational status.

This field is expressed in seconds and is an unsighed 32 bit value.

2.2.8 Status

Status provides a simple text string indicating whether a member proxy is currently able to handle requests (UP) or refused a connection when last contacted (DOWN).

2.2.9 Load Factor

Load Factor is a relative amount of the total load for an array that should be handled by any given member of the array.

Load Factor is specified as an integer and relative weight is computed against other integer values in the table.

2.2.10 Cache Size

Cache size is an informational field that indicates the size of the cache held by a particular member of an array.

Cache size is specified in Megabytes (MB) and represents the maximum potential size of a disk cache for this server.

3. Routing Function

Once an agent has a Proxy Array Membership Table. It uses a mathematical hash function to determine which of the members of the array should be the receptacle of a particular URL request.

This routing function involves constructing n "scores" using a hash of the request URL plus a hash of each of the k proxies in the Proxy Array Membership Table.

Both the URL and the proxy names are hashed in order to minimize the disruption of target routes if a member of the target array can't be contacted.

Hashes of the URL and proxy name are constructed using the algorithm described in 3.1 and combined using the algorithm described in 3.2.

3.1. Hash Function

The hash function outputs a 32 bit unsigned integers based on a zero-terminated ASCII input string. The machine name and domain names of the URL, the protocol, and the machine names of each member proxy should be evaluated in lower case since that portion of the URL is case insensitive.

Because irreversibility and strong cryptographic features are unnecessary for this application, a very simple and fast hash function based on the bitwise left rotate operator is used.

```
For (each char in URL):
      URL_Hash +=  _rotl(URL_Hash, 19) + char ;
```

Member proxy hashes are computed in a similar manner:

```
For (each char in MemberProxyName):
      MemberProxy_Hash +=  _rotl(MemberProxy_Hash, 19) + char ;
```

Becaues member names are often similar to each other, their hash values are further spread across hash space via the following additional operations:

```
MemberProxy_Hash += MemberProxy_Hash * 0x62531965 ;
MemberProxy_Hash = _rotl (MemberProxy_Hash, 21) ;
```

3.2. Hash Combination

Hashes are combined by first exclusive or-ing (XOR) the URL hash by the machine name and then multiplying by a constant and performing a bitwise rotation.

All final and intermediate values are 32 bit unsigned integers.

```
Combined_Hash = (URL_hash ^ MemberProxy_Hash) ;
Combined_Hash += Combined_Hash * 0x62531965 ;
Combined_Hash = _rotl(Combined_Hash, 21) ;
```

3.3. Load Factor

Support for array members with differing HTTP processing & caching capacity is achieved by multiplying each of the combined hash values by a Load Factor Multiplier.

The Load Factor Multiplier for an individual member is calculated by taking each member's relative Load Factor and applying the following formula:

The Load Factor Multiplier must be calculated from the smallest P_k to the largest P_k. The sum of all P_k's must be 1.

For each proxy server 1,...,K, the Load Factor Multiplier, X_k, is calculated iteratively as follows:

All X_n values are 32 bit floating point numbers.

```
X_1 = pow ((K*p_1), (1/K))

X_k = ([K-k+1] * [P_k - P_{k-1}])/(X_1 * X_2 * ... * X_{k-1})
X_k += pow ((X_{k-1}, {K-k+1})
X_k = pow (X_k, {1/(K-k+1)})
```

where:

X_k = Load Factor Multiplier for proxy k
K = number of proxies in an array
P_k = relative percent of the load that proxy k should handle

This is then combined with the previously computed hashes as

Resultant_value = Combined_Hash * X_k

3.4. Route Selection

The "score" for a particular combination of URL plus proxy is its resultant value. Once the agent determines the scores of the K proxies, it routes the URL query to the proxy with the highest score.

3.5. Member Failure Routing

If a proxy can not contact the designated member of a proxy array in order to forward an HTTP request, that proxy should route the request to the second highest scoring proxy in the target array.

4. Client-side implementation

CARP can be implemented on client-side HTTP browsers via the use of the Proxy AutoConfig file described in [1] and [2].

5. Versioning

If a downstream proxy receives an Array Membership Table with a greater version # than that proxy is able to parse, it should fall back to simple proxy request routing to any administrator defined upstream proxy server.

6. Security Considerations

This draft does not discuss relevant security considerations.

7. Open Issues

8. Acknowledgements

The author would like to thank Brian Smith, Kip Compton, and Kerry Schwartz for their assistance in preparing this document.

Most of the architecture & design of CARP stem from work conducted by Brian Smith at Microsoft Corp.

9. References

[1] Luotonen, Ari., "Navigator Proxy Auto-Config File Format", Netscape Corporation, http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html, March 1996.

[3] Wessels, Duane., "Internet Cache Protocol Version 2", http://ds.internic.net/internet-drafts/draft-wessels-icp-v2-00.txt, March 21, 1997.

[4] Sharp Corporation., "Super Proxy Script", http://naragw.sharp.co.jp/sps/, August 9, 1996.

[5] Fielding, R., et. al, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, UC Irvine, January 1997.

[6] Valloppillil & Cohen, "Hierarchical HTTP Routing Protocol", http://ircache.nlanr.net/Cache/ICP/draft-vinod-icp-traffic-dist-00.txt, April 21, 1997.

[7] Thaler, David & Ravishankar, Chinya. "Using Name-Based Mappings to Increase Hit Rates," ACM/IEEE Transactions on Networking. to appear.

10. Author Information

Vinod Valloppillil
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Phone: 1.206.703.3460
Email: VinodV@Microsoft.Com

Keith W. Ross
University of Pennsylvania
Department of Systems Engineering
Philadelphia, PA 19104

Phone: 1.215.898.6069
Email: Ross@UPenn.Edu