Network Working Group
Internet-Draft
Expires: January 2, 2001

I. Cooper
Mirror Image
I. Melve
UNINETT
G. Tomlinson
Novell

July 4, 2000

Internet Web Replication and Caching Taxonomy
draft-ietf-wrec-taxonomy-05.txt

Status of this Memo

Copyright Notice

Abstract

This memo specifies standard terminology and the current taxonomy of web replication and caching infrastructure deployed today. It introduces standard concepts and protocols used today within this application domain. Currently deployed solutions employing these technologies are presented to establish a standard taxonomy. Known problems with caching proxies are covered in an accompanying document[23], and are not part of this document. This document presents open protocols and points to published material for each protocol.

Table of Contents

1. Introduction

Since its introduction in 1990, the World-Wide Web has evolved from a simple client server model into a sophisticated distributed architecture. This evolution has been driven largely due to the scaling problems associated with exponential growth. Distinct paradigms and solutions have emerged to satisfy specific requirements. Two core infrastructure components being employed to meet the demands of this growth are replication and caching. In many cases, there is a need for web caches and replicated services to be able to coexist.

There are many protocols, both open and proprietary, employed in web replication and caching today. A majority of the open protocols include DNS[15], Cache Digests[17][19], CARP[4], HTTP[1], ICP[5], PAC[2], SOCKS[14], WPAD[3], and WCCP[13]. Additional protocols are being planned to address emerging solution requirements.

This memo specifies standard terminology and the taxonomy of web replication and caching infrastructure deployed in the Internet today. The principal goal of this document is to establish a common understanding and reference point of this application domain.

We also expect that this document will be used in the creation of a standard architectural framework for efficient, reliable, and predictable service in a web which includes both replicas and caches.

2. Terminology

The following terminology provides definitions of common terms used within the web replication and caching community. Base terms are taken, where possible, from the HTTP/1.1 specification[1] and are included here for reference. First- and second-order derivatives are constructed from these base terms to help define the relationships that exist within this area.

Terms that are in common usage and which are contrary to definitions in RFC2616 and this document are highlighted.

2.1 Base Terms

The majority of these terms are taken as-is from RFC2616[1], and are included here for reference.

client (taken from [1])
A program that establishes connections for the purpose of sending requests.

server (taken from [1])
An application program that accepts connections in order to service requests by sending back responses. Any given program may be capable of being both a client and a server; our use of these terms refers only to the role being performed by the program for a particular connection, rather than to the program's capabilities in general. Likewise, any server may act as an origin server, proxy, gateway, or tunnel, switching behavior based on the nature of each request.

proxy (taken from [1])
An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, with possible translation, to other servers. A proxy MUST implement both the client and server requirements of this specification. A "transparent proxy" is a proxy that does not modify the request or response beyond what is required for proxy authentication and identification. A "non-transparent proxy" is a proxy that modifies the request or response in order to provide some added service to the user agent, such as group annotation services, media type transformation, protocol reduction, or anonymity filtering. Except where either transparent or non-transparent behavior is explicitly stated, the HTTP proxy requirements apply to both types of proxies.

Note: The term "transparent proxy" refers to a semantically transparent proxy as described in [1], not what is commonly

understood within the caching community. We recommend that the term "transparent proxy" is always prefixed to avoid confusion (e.g. "network transparent proxy"). However, see definition of "interception proxy" below.

The above condition requiring implementation of both the server and client requirements of HTTP/1.1 is only appropriate for a non-network transparent proxy.

cache (taken from [1])
A program's local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, though a cache cannot be used by a server that is acting as a tunnel.

Note: The term "cache" used alone often is meant as "caching proxy".

Note: There are additional motivations for caching, for example reducing server load (as a further means to reduce response time).

cacheable (taken from [1])
A response is cacheable if a cache is allowed to store a copy of the response message for use in answering subsequent requests. The rules for determining the cacheability of HTTP responses are defined in section 13. Even if a resource is cacheable, there may be additional constraints on whether a cache can use the cached copy for a particular request.

gateway (taken from [1])
A server which acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the origin server for the requested resource; the requesting client may not be aware that it is communicating with a gateway.

tunnel (taken from [1])
An intermediary program which is acting as a blind relay between two connections. Once active, a tunnel is not considered a party to the HTTP communication, though the tunnel may have been initiated by an HTTP request. The tunnel ceases to exist when both ends of the relayed connections are closed.

replication (definition from FOLDOC[22])
Creating and maintaining a duplicate copy of a database or file system on a different computer, typically a server.

inbound/outbound (taken from [1])
Inbound and outbound refer to the request and response paths for

messages: "inbound" means "traveling toward the origin server", and "outbound" means "traveling toward the user agent".

network element
A network device that introduces multiple paths between source and destination, transparent to HTTP.

2.2 First order derivative terms

The following terms are constructed taking the above base terms as foundation.

origin server (taken from [1])
The server on which a given resource resides or is to be created.

user agent (taken from [1])
The client which initiates a request. These are often browsers, editors, spiders (web-traversing robots), or other end user tools.

caching proxy
A proxy with a cache, acting as a server to clients, and a client to servers.

Caching proxies are often referred to as "proxy caches" or simply "caches". The term "proxy" is also frequently misused when referring to caching proxies.

surrogate
A gateway co-located with an origin server, or at a different point in the network, delegated the authority to operate on behalf of, and typically working in close co-operation with, one or more origin servers. Responses are typically delivered from an internal cache.

Surrogates may derive cache entries from the origin server or from another of the origin server's delegates. In some cases a surrogate may tunnel such requests.

Where close co-operation between origin servers and surrogates exists, this enables modifications of some protocol requirements, including the Cache-Control directives in [1]. Such modifications have yet to be fully specified.

Devices commonly known as "reverse proxies" and "(origin) server accelerators" are both more properly defined as surrogates.

reverse proxy
See "surrogate".

server accelerator
See "surrogate".

2.3 Second order derivatives

The following terms further build on first order derivatives:

master origin server
An origin server on which the definitive version of a resource resides.

replica origin server
An origin server holding a replica of a resource, but which may act as an authoritative reference for client requests.

content consumer
The user or system that initiates inbound requests, through use of a user agent.

browser
A special instance of a user agent that acts as a content presentation device for content consumers.

2.4 Topological terms

The following definitions are added to describe caching device topology:

user agent cache
The cache within the user agent program.

local caching proxy
The caching proxy to which a user agent connects.

intermediate caching proxy
Seen from the content consumer's view, all caches participating in the caching mesh that are not the user agent's local caching proxy.

cache server
A server to requests made by local and intermediate caching proxies, but which does not act as a proxy.

cache array
A cluster of caching proxies, acting logically as one service and partitioning the resource name space across the array. Also known as "diffused array" or "cache cluster".

caching mesh

a loosely coupled set of co-operating proxy- and (optionally) caching-servers, or clusters, acting independently but sharing cacheable content between themselves using inter-cache communication protocols.

2.5 Automatic use of proxies

Network administrators may wish to force or facilitate the use of proxies by clients, enabling such configuration within the network itself or within automatic systems in user agents, such that the content consumer need not be aware of any such configuration issues.

The terms that describe such configurations are given below.

automatic user-agent proxy configuration
: The technique of discovering the availability of one or more proxies and the automated configuration of the user agent to use them. The use of a proxy is transparent to the content consumer but not to the user agent. The term "automatic proxy configuration" is also used in this sense.

traffic interception
: The process of using a network element to examine network traffic to determine whether it should be redirected.

traffic redirection
: Redirection of client requests from a network element performing traffic interception to a proxy. Used to deploy (caching) proxies without the need to manually reconfigure individual user agents, or to force the use of a proxy where such use would not otherwise occur.

interception proxy (a.k.a. "transparent proxy", "transparent cache")
: The term "transparent proxy" has been used within the caching community to describe proxies used with zero configuration within the user agent. Such use is somewhat transparent to user agents. Due to discrepancies with [1] (see definition of "proxy" above), and objections to the use of the word "transparent", we introduce the term "interception proxy" to describe proxies that receive redirected traffic flows from network elements performing traffic interception.

  Interception proxies receive inbound traffic flows through the process of traffic redirection. (Such proxies are deployed by network administrators to facilitate or require the use of appropriate services offered by the proxy). Problems associated with the deployment of interception proxies are described in the companion document "Known HTTP Proxy/Caching Problems"[23]. The use of interception proxies requires zero configuration of the

user agent which act as though communicating directly with an origin server.

3. Distributed System Relationships

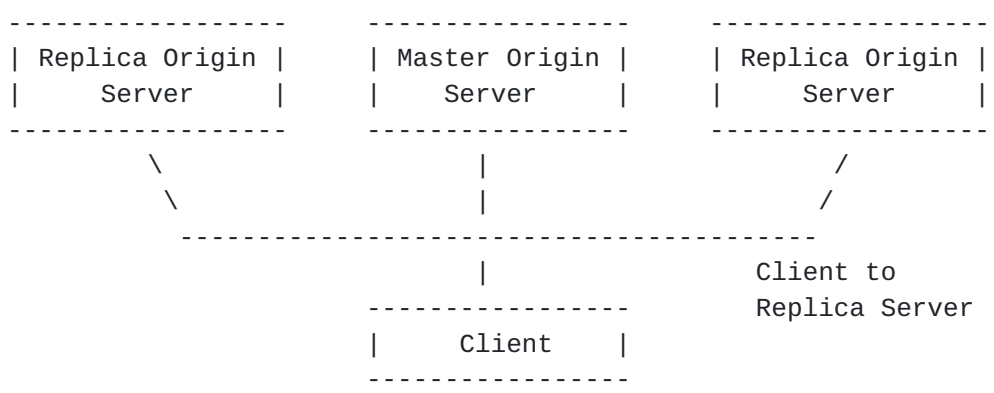
This section identifies the relationships that exist in a distributed replication and caching environment. Having defined these relationships, later sections describe the communication protocols used in each relationship.

3.1 Replication Relationships

The following sections describe relationships between clients and replicas and between replicas themselves.

3.1.1 Client to Replica

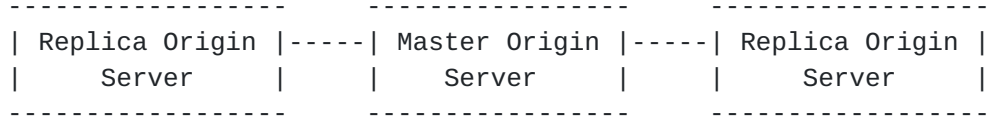
A client may communicate with one or more replica origin servers, as well as with master origin servers. (In the absence of replica servers the client interacts directly with the origin server as is the normal case.)



Protocols used to enable the client to use one of the replicas can be found in Section 4.

3.1.2 Inter-Replica

This is the relationship between master origin server(s) and replica origin servers, to replicate data sets that are accessed by clients in the relationship shown in Section 3.1.1.



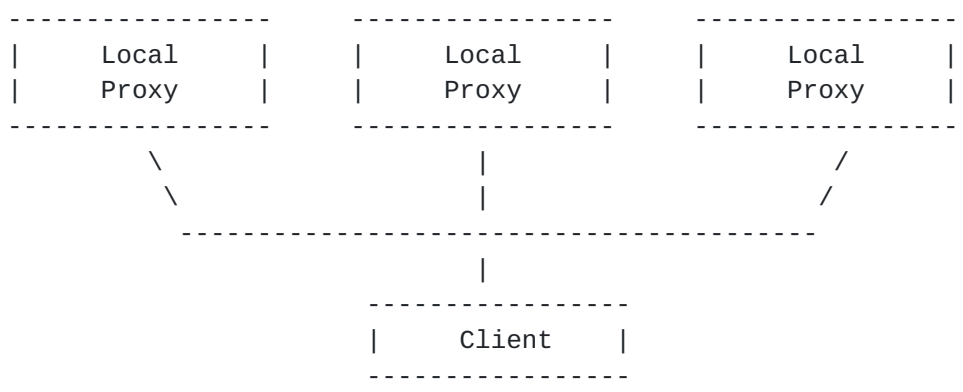
Protocols used in this relationship can be found in Section 5.

3.2 Proxy Relationships

There are a variety of ways in which (caching) proxies and cache servers communicate with each other, and with user agents.

3.2.1 Client to Non-Interception Proxy

A client may communicate with zero or more proxies for some or all requests. Where the result of communication results in no proxy being used, the relationship is between client and (replica) origin server (see Section 3.1.1).

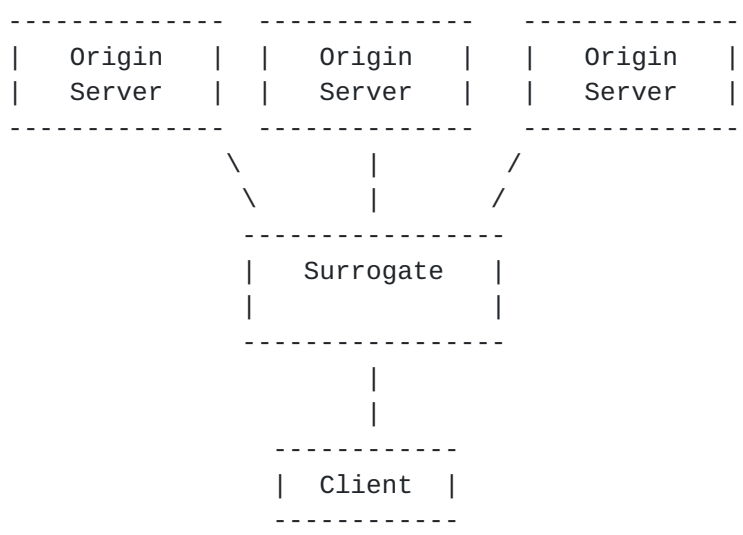


In addition, a user agent may interact with an additional server - operated on behalf of a proxy for the purpose of automatic user agent proxy configuration.

Schemes and protocols used in these relationships can be found in Section 6.

3.2.2 Client to Surrogate to Origin Server

A client may communicate with zero or more surrogates for requests intended for one or more origin servers. Where a surrogate is not used, the client communicates directly with an origin server. Where a surrogate is used the client communicates as if with an origin server. The surrogate fulfills the request from its internal cache, or acts as a gateway or tunnel to the origin server.

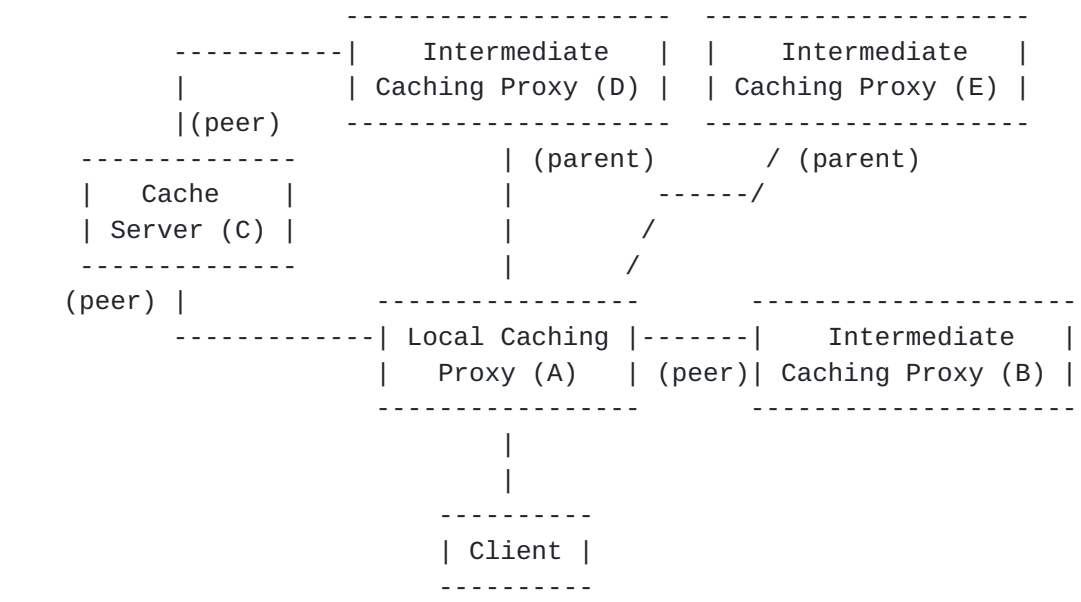


3.2.3 Inter-Proxy

Inter-Proxy relationships exist as meshes (loosely coupled) and clusters (tightly coupled).

3.2.3.1 (Caching) Proxy Meshes

Within a loosely coupled mesh of (caching) proxies, communication can happen at the same level between peers, and with one or more parents.



Client included for illustration purposes only

An inbound request may be routed to one of a number of intermediate

(caching) proxies based on a determination of whether that parent is

better suited to resolving the request.

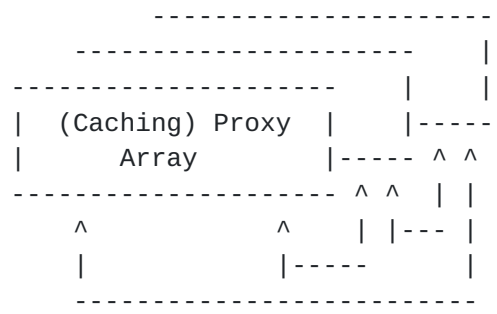
For example, in the above figure, Cache Server C and Intermediate Caching Proxy B are peers of the Local Caching Proxy A, and may only be used when the resource requested by A already exists on either B or C. Intermediate Caching Proxies D & E are parents of A, and it is A's choice of which to use to resolve a particular request.

The relationship between A & B only makes sense in a caching environment, while the relationships between A & D and A & E are also appropriate where D or E are non-caching proxies.

Protocols used in these relationships can be found in Section 7.1.

3.2.3.2 (Caching) Proxy Arrays

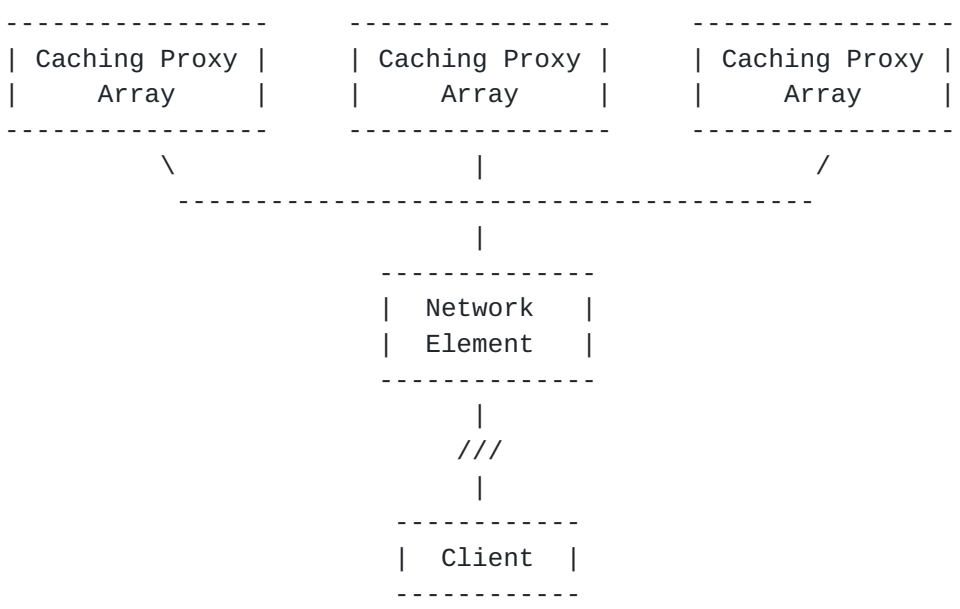
Where a user agent may have a relationship with a proxy, it is possible that it may instead have a relationship with an array of proxies arranged in a tightly coupled mesh.



Protocols used in this relationship can be found in Section 7.2.

3.2.4 Network Element to Caching Proxy

A network element performing traffic interception may choose to redirect requests from a client to a specific proxy within an array. (It may also choose not to redirect the traffic, in which case the relationship is between client and (replica) origin server, see Section 3.1.1.)



The interception proxy may be directly in-line of the flow of traffic - in which case the intercepting network element and interception proxy form parts of the same hardware system - or may be out-of-path, requiring the intercepting network element to redirect traffic to another network segment. In this latter case, communication protocols enable the intercepting network element to stop and start redirecting traffic when the interception proxy becomes (un)available. Details of these protocols can be found in Section 8.

4. Replica Selection

This section describes the schemes and protocols used in the cooperation and communication between client and replica origin web servers. The ideal situation is to discover an optimal replica origin server for clients to communicate with. Optimality is a policy based decision, often based upon proximity, but may be based on other criteria such as load.

4.1 Navigation Hyperlinks

Authoritative reference:
This memo.

Description:
The simplest of client to replica communication mechanisms. This utilizes hyperlink URIs embedded in web pages that point to the individual replica origin servers. The content consumer manually selects the link of the replica origin server they wish to use.

Security:
Relies on the protocol security associated with the appropriate URI scheme.

Deployment:
Probably the most commonly deployed client to replica communication mechanism. Ubiquitous interoperability with humans.

Submitter:
Document editors.

4.2 HTTP Redirection

Authoritative reference:
This memo.

Description:
A simple and commonly used mechanism to connect clients with replica origin servers is to use HTTP redirection. Clients are redirected to an optimal replica origin server via the use of the HTTP[1] protocol response codes, e.g. 302 "Found", or 307 "Temporary Redirect". A client establishes HTTP communication with one of the replica origin servers. The initially contacted replica origin server can then either choose to accept the service or redirect the client again. Refer to section 10.3 in HTTP/1.1[1] for information on HTTP response codes.

Security:
Relies entirely upon HTTP security.

Deployment:
Observed at a number of large web sites. Extent of usage in the Internet is unknown.

Submitter:
Document editors.

4.3 DNS Redirection

Authoritative reference:

* RFC1794 DNS Support for Load Balancing Proximity[15]

* This memo

Description:
The Domain Name Service (DNS) provides a more sophisticated client to replica communication mechanism. This is accomplished by DNS servers that sort resolved IP addresses based upon quality of service policies. When a client resolves the name of an origin server, the enhanced DNS server sorts the available IP addresses of the replica origin servers starting with the most optimal replica and ending with the least optimal replica.

Security:
Relies entirely upon DNS security, and other protocols that may be used in determining the sort order.

Deployment:
Observed at a number of large web sites and large ISP web hosted services. Extent of usage in the Internet is unknown, but is believed to be increasing.

Submitter:
Document editors.

5. Inter-Replica Communication

This section describes the cooperation and communication between master- and replica- origin servers. Used in replicating data sets between origin servers.

5.1 Batch Driven Replication

Authoritative reference:
This memo.

Description:
The replica origin server to be updated initiates communication with a master origin server. The communication is established at intervals based upon queued transactions which are scheduled for deferred processing. The scheduling mechanism policies vary, but generally are re-occurring at a specified time. Once communication is established, data sets are copied to the initiating replica origin server.

Security:
Relies upon the protocol being used to transfer the data set. FTP[10] and RDIST are the most common protocols observed.

Deployment:
Very common for synchronization of mirror sites in the Internet.

Submitter:
Document editors.

5.2 Demand Driven Replication

Authoritative reference:
This memo.

Description:
Replica origin servers acquire content as needed due to client demand. When a client requests a resource that is not in the data set of the replica origin server/surrogate, an attempt is made to resolve the request by acquiring the resource from the master origin server, returning it to the requesting client.

Security:
Relies upon the protocol being used to transfer the resources. FTP[10], Gopher[11], HTTP[1] and ICP[5] are the most common protocols observed.

Deployment:
Observed at several large web sites. Extent of usage in the

Internet is unknown.

Submitter:
Document editors.

5.3 Synchronized Replication

Authoritative reference:
This memo.

Ed note: there is no IETF protocol specified at this time. The editors are aware of at least two open source protocols, AFS and CODA, along with one expired IETF draft <draft-leach-cifs-v1-spec-01.txt> and one proprietary protocol Novell NRS; none of which can be considered an authoritative reference.

Description:
Replicated origin servers cooperate using synchronized strategies and specialized replica protocols to keep the replica data sets coherent. Synchronization strategies range from tightly coherent (a few minutes) to loosely coherent (a few or more hours). Updates occur between replicas based upon the synchronization time constraints of the coherency model employed and are generally in the form of deltas only.

Security:
All of the known protocols utilize strong cryptographic key exchange methods, which are either based upon the Kerberos shared secret model or the public/private key RSA model.

Deployment:
Observed at a few sites, primarily at university campuses.

Submitter:
Document editors.

6. User Agent to Proxy Configuration

This section describes the configuration, cooperation and communication between user agents and proxies.

6.1 Manual Proxy Configuration

Authoritative reference:
This memo.

Description:
Each user must configure her user agent by supplying information pertaining to proxied protocols and local policies.

Security:
The potential for doing wrong is high; each user individually sets preferences.

Deployment:
Widely deployed, used in all current browsers. Most browsers also support additional options.

Submitter:
Document editors.

6.2 Proxy Auto Configuration (PAC)

Authoritative reference:
No RFC, no Internet-Draft; Navigator Proxy Auto-Config File Format[2].

Description:
A JavaScript script retrieved from a web server is executed for each URL accessed to determine the appropriate proxy (if any) to be used to access the resource. User agents must be configured to request this script upon startup. There is no bootstrap mechanism, manual configuration is necessary.

Despite manual configuration, the process of proxy configuration is simplified by centralizing it within a script at a single location.

Security:
Common policy per organization possible but still requires initial manual configuration. PAC is better than "manual proxy configuration" since PAC administrators may update the proxy configuration without further user intervention.

Interoperability of PAC files is not high, since different

browsers have slightly different interpretations of the same script, possibly leading to undesired effects.

Deployment:
Implemented in Netscape Navigator and Microsoft Internet Explorer.

Submitter:
Document editors.

6.3 Cache Array Routing Protocol (CARP) v1.0

Authoritative reference:
Expired Internet-Draft: draft-vinod-carp-v1-03.txt[4]

Note: Reference kept since there is known implementation.

Description:
User agents may use CARP directly as a hash function based proxy selection mechanism. They need to be configured with the location of the cluster information.

Security:
Security considerations are not covered in the specification drafts.

Deployment:
Implemented in Microsoft Proxy Server, Squid. Implemented in user agents via PAC scripts.

Submitter:
Document editors.

6.4 Web Proxy Auto-Discovery Protocol (WPAD)

Authoritative reference:
Expired Internet-Draft: draft-ietf-wrec-wpad-01.txt[3]

Description:
WPAD uses a collection of pre-existing Internet resource discovery mechanisms to perform web proxy auto-discovery.

The only goal of WPAD is to locate the PAC URL[2]. WPAD does not specify which proxies will be used. WPAD supplies the PAC URL, and the PAC script then operates as defined above to choose proxies per resource request.

The WPAD protocol specifies the following:

* how to use each mechanism for the specific purpose of web

proxy auto-discovery

* the order in which the mechanisms should be performed

* the minimal set of mechanisms which must be attempted by a WPAD compliant user agent

The resource discovery mechanisms utilized by WPAD are as follows:

* Dynamic Host Configuration Protocol DHCP

* Service Location Protocol SLP

* "Well Known Aliases" using DNS A records

* DNS SRV records

* "service: URLs" in DNS TXT records

Security:
Relies upon DNS and HTTP security.

Deployment:
Implemented in user agents and caching proxy servers. More than two independent implementations.

Submitter:
Josh Cohen, Microsoft, joshco@microsoft.com

7. Inter-Proxy Communication

7.1 Loosely coupled Inter-Proxy Communication

This section describes the cooperation and communication between caching proxies.

7.1.1 Internet Cache Protocol (ICP)

Authoritative reference:
RFC2186 Internet Cache Protocol (ICP), version 2[5]

Description:
ICP is used by proxies to query other (caching) proxies about web resources, to see if the requested resource is present on the other system.

ICP uses UDP. Since UDP is an uncorrected network transport protocol, an estimate of network congestion and availability may be calculated by ICP loss. This rudimentary loss measurement provides, together with round trip times, a load balancing method for caches.

Security:
See RFC2187[6]

ICP does not convey information about HTTP headers associated with resources. HTTP headers may include access control and cache directives. Since proxies ask for the availability of resources, and subsequently retrieve them using HTTP, false cache hits may occur (object present in cache, but not accessible to a sibling is one example).

ICP suffers from all the security problems of UDP.

Deployment:
Widely deployed. Most current caching proxy implementations support ICP in some form.

Submitter:
Document editors.

See also Internet-Draft draft-lovric-icp-ext-02.txt[7], ICP development Web page[8], ICP1.4 specification[9].

7.1.2 Hyper Text Caching Protocol

Authoritative reference:
RFC2756 Hyper Text Caching Protocol (HTCP/0.0)[18]

Description:
HTCP is a protocol for discovering HTTP caching proxies and cached data, managing sets of HTTP caching proxies, and monitoring cache activity.

HTCP requests include HTTP header material, while ICPv2 does not, enabling HTCP replies to more accurately describe the behaviour that would occur as a result of a subsequent HTTP request for the same resource.

Security:
Optionally uses HMAC-MD5[20] shared secret authentication. Protocol is subject to attack if authentication is not used.

Deployment:
HTCP is implemented in Squid[24] and the Web Gateway Interceptor[25].

Submitter:
Document editors.

7.1.3 Cache Digest

Authoritative reference:

* No RFC, no Internet-Draft; Cache Digest specification - version 5[17]

* Summary Cache[19](see note)

Description:
Cache Digests are a response to the problems of latency and congestion associated with previous inter-cache communication mechanisms such as the Internet Cache Protocol (ICP)[5] and the Hyper Text Cache Protocol[18]. Unlike these protocols, Cache Digests support peering between caching proxies and cache servers without a request-response exchange taking place for each inbound request. Instead, a summary of the contents in cache (the Digest) is fetched by other systems that peer with it. Using Cache Digests it is possible to determine with a relatively high degree of accuracy whether a given resource is cached by a particular system.

Cache Digests are both an exchange protocol and a data format [17]

Security:
If the contents of a Digest are sensitive, they should be protected. Any methods which would normally be applied to secure an HTTP connection can be applied to Cache Digests.

A 'Trojan horse' attack is currently possible in a mesh: System A A can build a fake peer Digest for system B and serve it to B's peers if requested. This way A can direct traffic toward/from B. The impact of this problem is minimized by the 'pull' model of transferring Cache Digests from one system to another.

Cache Digests provide knowledge about peer cache content on a URL level. Hence, they do not dictate a particular level of policy management and can be used to implement various policies on any level (user, organization, etc.).

Deployment:
Cache Digests are supported in Squid.

Cache Meshes:

* NLANR Mesh[26]

* TF-CACHE mesh[27] (European Academic networks)

Submitter:
Alex Rousskov, NLANR, rousskov@nlanr.net for [17]
Pei Cao for [19]

Note: The technology of Summary Cache[19]is patent pending by the University of Wisconsin-Madison.

7.1.4 Cache Pre-filling

Authoritative reference:
Internet-Draft: draft-lovric-francetelecom-satellites-01.txt[16]

Description:
Cache pre-filling is a push-caching implementation. It is particularly well adapted to IP-multicast networks because it allows preselected resources to be simultaneously inserted into caches within the targeted multicast group. Different implementations of cache pre-filling already exist, especially in satellite contexts. However, there is still no standard for this kind of push-caching and vendors propose solutions either based on dedicated equipment or public domain caches extended with a pre-filling module.

Security:
Relies on the inter-cache protocols being employed.

Deployment:
Observed in two commercial content distribution service providers.

Submitter:
Ivan Lovric, France Telecom, ivan.lovric@cnet.francetelecom.fr

7.2 Tightly Coupled Inter-Cache Communication

7.2.1 Cache Array Routing Protocol (CARP) v1.0

Also see Section 6.3

Authoritative reference:
Expired Internet-Draft: draft-vinod-carp-v1-03.txt[4]

Note: Reference kept since there is known deployment.

Description:
CARP is a hashing function for dividing URL-space among a cluster of proxies. Included in CARP is the definition of a Proxy Array Membership Table, and ways to download this information.

A user agent which implements CARP v1.0 can allocate and intelligently route requests for the 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 may be improved.

Security:
Security considerations are not covered in the specification drafts.

Deployment:
Implemented in caching proxy servers. More than two independent implementations.

Submitter:
Document editors.

8. Network Element Communication

This section describes the cooperation and communication between proxies and network elements. Examples of such network elements include routers and switches. Generally used for deploying interception proxies and/or diffused arrays.

8.1 Web Cache Control Protocol (WCCP)

Authoritative reference:
Expired Internet-Draft: draft-ietf-wrec-web-pro-00.txt[13]

Description:
WCCP V1 runs between a router functioning as a redirecting network element and out-of-path interception proxies. The protocol allows one or more proxies to register with a single router to receive redirected traffic. It also allows one of the proxies, the designated proxy, to dictate to the router how redirected traffic is distributed across the array.

Security:
WCCP V1 has no security features.

Deployment:
Network elements: WCCP V1 is deployed on a wide range of Cisco routers.
Caching proxies: WCCP V1 is deployed on a number of vendors' caching proxies.

Submitter:
David Forster, CISCO, dforster@cisco.com

8.2 Network Element Control Protocol (NECP)

Authoritative reference:
draft-cerpa-necp-02.txt[21]

Description:
NECP provides methods for network elements to learn about server capabilities, availability, and hints as to which flows can and cannot be serviced. This allows network elements to perform load balancing across a farm of servers, redirection to interception proxies, and cut-through of flows that cannot be served by the farm.

Security:
Optionally uses HMAC-SHA-1[20] shared secret authentication along with complex sequence numbers to provide moderately strong security. Protocol is subject to attack if authentication is not

used.

Deployment:
NECP is a new protocol being implemented by more than two network element vendors and by more than two caching proxy vendors. It is anticipated to be broadly deployed in the Internet during the year 2000.

Submitter:
Gary Tomlinson, Novell, garyt@novell.com

8.3 SOCKS

Authoritative reference:
RFC1928 SOCKS Protocol Version 5[14]

Description:
SOCKS is primarily used as a caching proxy to firewall protocol. Although firewalls don't conform to the narrowly defined network element definition above, they are a integral part of the network infrastructure. When used in conjunction with a firewall, SOCKS provides a authenticated tunnel between the caching proxy and the firewall.

Security:
An extensive framework provides for multiple authentication methods. Currently, SSL, CHAP, DES, 3DES are known to be available.

Deployment:
SOCKS is been widely deployed in the Internet.

Submitter:
Document editors.

9. Security Considerations

This document provides a taxonomy for web caching and replication. Recommended practice, architecture and protocols are not described in detail.

By definition, replication and caching involve the copying of resources. There are legal implications of making and keeping transient or permanent copies; these are not covered here.

Information on security of each protocol referred to by this memo is provided in the preceding sections, and in their accompanying documentation. HTTP security is discussed in section 15 of RFC2616[1], the HTTP/1.1 specification, and to a lesser extent in RFC1945[12], the HTTP/1.0 specification. RFC2616 contains security considerations for HTTP proxies.

Caching proxies have the same security issues as other application level proxies. Application level proxies are not covered in these security considerations. IP number based authentication is problematic when a proxy is involved in the communications. Details are not discussed here.

9.1 Authentication

Requests for web resources, and responses to such requests, may be directed to replicas and/or may flow through intermediate proxies. The integrity of communication needs to be preserved to ensure protection from both loss of access and from unintended change.

9.1.1 Man in the middle attacks

HTTP proxies are men-in-the-middle, the perfect place for a man-in-the-middle-attack. A discussion of this is found in section 15 of RFC2616[1].

9.1.2 Trusted third party

A proxy must either be trusted to act on behalf of the origin server and/or client, or it must act as a tunnel. When presenting cached objects to clients, the clients need to trust the caching proxy to act on behalf on the origin server.

A replica may get accreditation from the origin server.

9.1.3 Authentication based on IP number

Authentication based on the client's IP number is problematic when connecting through a proxy, since the authenticating device only has

access to the proxy's IP number. One (problematic) solution to this is for the proxy to spoof the client's IP number for inbound requests.

Authentication based on IP number assumes that the end-to-end properties of the Internet are preserved. This is typically not the case for environments containing interception proxies.

9.2 Privacy

9.2.1 Trusted third party

When using a replication service, one must trust both the replica origin server and the replica selection system.

Redirection of traffic - either by automated replica selection methods, or within proxies - may introduce third parties the end user and/or origin server must to trust. In the case of interception proxies, such third parties are often unknown to both end points of the communication. Unknown third parties may have security implications.

Both proxies and replica selection services may have access to aggregated access information. A proxy typically knows about accesses by each client using it, information that is more sensitive than the information held by a single origin server.

9.2.2 Logs and legal implications

Logs from proxies should be kept secure, since they provide information about users and their patterns of behaviour. A proxy's log is even more sensitive than a web server log, as every request from the user population goes through the proxy. Logs from replica origin servers may need to be amalgamated to get aggregated statistics from a service, and transporting logs across borders may have legal implications. Log handling is restricted by law in some countries.

Requirements for object security and privacy are the same in a web replication and caching system as it is in the Internet at large. The only reliable solution is strong cryptography. End-to-end encryption frequently makes resources uncacheable, as in the case of SSL encrypted web sessions.

9.3 Service security

9.3.1 Denial of service

Any redirection of traffic is susceptible to denial of service attacks at the redirect point, and both proxies and replica selection services may redirect traffic.

By attacking a proxy, access to all servers may be denied for a large set of clients.

It has been argued that introduction of an interception proxy is a denial of service attack, since the end-to-end nature of the Internet is destroyed without the content consumer's knowledge.

9.3.2 Replay attack

A caching proxy is by definition a replay attack.

9.3.3 Stupid configuration of proxies

It is quite easy to have a stupid configuration which will harm service for content consumers. This is the most common security problem with proxies.

9.3.4 Copyrighted transient copies

The legislative forces of the world are considering the question of transient copies, like those kept in replication and caching system, being legal. The legal implications of replication and caching are subject to local law.

Caching proxies need to preserve the protocol output, including headers. Replication services need to preserve the source of the objects.

9.3.5 Application level access

Caching proxies are application level components in the traffic flow path, and may give intruders access to information that was previously only available at the network level in a proxy-free world. Some network level equipment may have required physical access to get sensitive information. Introduction of application level components may require additional system security.

10. Acknowledgements

The editors would like to thank the following for their assistance: David Forster, Alex Rousskov, Josh Cohen, John Martin, John Dilley, Ivan Lovric, Joe Touch, Henrik Nordstrom, Patrick McManus, Duane Wessels, Wojtek Sylwestrzak, Ted Hardie, Misha Rabinovich, Larry Masinter, Keith Moore, Roy Fielding, Patrick Faltstrom, Hilarie Orman, Mark Nottingham and Oskar Batuner.

References

[1] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, <URL:http://www.ietf.org/rfc/rfc2616.txt>.

[2] Netscape, Inc., "Navigator Proxy Auto-Config File Format", March 1996, <URL:http://www.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html>.

[3] Gauthier, P., Cohen, J., Dunsmuir, M. and C. Perkins, "The Web Proxy Auto-Discovery Protocol", draft-ietf-wrec-wpad-01.txt (work in progress), July 1999, <URL:http://www.wrec.org/Drafts/draft-ietf-wrec-wpad-01.txt>.

[4] Valloppillil, V. and K.W. Ross, "Cache Array Routing Protocol", draft-vinod-carp-v1-03.txt (work in progress), February 1998, <URL:http://www.wrec.org/Drafts/draft-vinod-carp-v1-03.txt>.

[5] Wessels, D. and K. Claffy, "Internet Cache Protocol (ICP), Version 2", RFC 2186, September 1997, <URL:http://www.ietf.org/rfc/rfc2186.txt>.

[6] Wessels, D. and K. Claffy, "Application of Internet Cache Protocol (ICP), Version 2", RFC 2187, September 1997, <URL:http://www.ietf.org/rfc/rfc2187.txt>.

[7] Lovric, I., "Internet Cache Protocol Extension", draft-lovric-icp-ext-02.txt (work in progress), October 1999, <URL:http://www.wrec.org/Drafts/draft-lovric-icp-ext-02.txt>.

[8] Wessels, D., "ICP Home Page", July 1999, <URL:http://ircache.nlanr.net/Cache/ICP/>.

[9] University of Southern California and University of Colorado-Boulder, "Internet Cache Protocol Specification 1.4", September 1994, <URL:http://excalibur.usc.edu/icpdoc/icp.html>.

[10] Postel, J. and J.K. Reynolds, "File Transfer Protocol (FTP)", RFC 959, Oct 1985, <URL:http://www.ietf.org/rfc/rfc0959.txt>.

[11] Anklesaria, F., McCahill, M., Lindner, P., Johnson, D., Torrey, D. and B. Alberti, "The Internet Gopher Protocol", RFC 1436, Mar 1993,

<URL:http://www.ietf.org/rfc/rfc1436.txt>.

[12] Berners-Lee, T., Fielding, R. and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996, <URL:http://www.ietf.org/rfc/rfc1945.txt>.

[13] Cieslak, M. and D. Forster, "Cisco Web Cache Control Protocol V1.0", draft-ietf-wrec-web-pro-00.txt (work in progress), June 1999, <URL:http://www.wrec.org/Drafts/draft-ietf-wrec-web-pro-00.txt>.

[14] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D. and L. Jones, "SOCKS Protocol Version 5", RFC 1928, March 1996, <URL:http://www.ietf.org/rfc/rfc1928.txt>.

[15] Brisco, T., "DNS Support for Load Balancing", RFC 1794, April 1995, <URL:http://www.ietf.org/rfc/rfc1794.txt>.

[16] Goutard, C., Lovric, I. and E. Maschio-Esposito, "Pre-filling a cache - A satellite overview", draft-lovric-francetelecom-satellites-01.txt (work in progress), February 2000, <URL:http://www.ietf.org/internet-drafts/draft-lovric-francetelecom-satellites-01.txt>.

[17] Hamilton, M., Rousskov, A. and D. Wessels, "Cache Digest specification - version 5", December 1998, <URL:http://www.squid-cache.org/CacheDigest/cache-digest-v5.txt>.

[18] Vixie, P. and D. Wessels, "Hyper Text Caching Protocol (HTCP/0.0)", RFC 2756, January 2000, <URL:http://www.ietf.org/rfc/rfc2756.txt>.

[19] Fan, L., Cao, P., Almeida, J. and A. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol", Proceedings of ACM SIGCOMM'98 pp. 254-265, September 1998, <URL:http://www.cs.wisc.edu/~cao/papers/summarycache.html>.

[20] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997, <URL:http://www.ietf.org/rfc/rfc2104.txt>.

[21] Cerpa, A., Elson, J., Beheshti, H., Chankhunthod, A., Danzig, P., Jalan, R., Neerdaels, C., Shroeder, T. and G. Tomlinson, "NECP: The Network Element Control Protocol", draft-cerpa-necp-02.txt (work in progress), February 2000, <URL:http://www.ietf.org/internet-drafts/draft-cerpa-necp-02.tx

t>.

[22] FOLDOC, "Free Online Dictionary of Computing: Replication", December 1997, <URL:http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?replication>.

[23] Cooper, I. and J. Dilley, "Known HTTP Proxy/Caching Problems", draft-ietf-wrec-known-prob-02.txt (work in progress), July 2000, <URL:http://www.ietf.org/internet-drafts/draft-ietf-wrec-known-prob-02.txt>.

[24] <URL:http://www.squid-cache.org/>

[25] <URL:http://www.vix.com/vix/wgi.html>

[26] <URL:http://ircache.nlanr.net/Cache/>

[27] <URL:http://www.terena.nl/task-force/tf-cache/>

Authors' Addresses

Ian Cooper
Mirror Image Internet, Inc.
49 Dragon Court
Woburn, MA 01801
USA

Phone: +1 781 376 1109
EMail: ian.cooper@mirror-image.com

Ingrid Melve
UNINETT
Tempeveien 22
Trondheim N-7465
Norway

Phone: +47 73 55 79 07
EMail: Ingrid.Melve@uninett.no

Gary Tomlinson
Novell Inc.
122 East 1700 South
Provo, Utah 84606
USA

Phone: +1 801 861 7021
EMail: garyt@novell.com

Full Copyright Statement

Acknowledgement

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