                    Known HTTP Proxy/Caching Problems

Status of this Memo

   This memo provides information for the Internet community. It does
   not specify an Internet standard of any kind. Distribution of this
   memo is unlimited.

Copyright Notice

Abstract

   This document catalogs a number of known problems with World Wide Web
   (WWW) (caching) proxies and cache servers. The goal of the document
   is to provide a discussion of the problems and proposed workarounds,
   and ultimately to improve conditions by illustrating problems. The
   construction of this document is a joint effort of the Web caching
   community.

Table of Contents

1. Introduction

   This memo discusses problems with proxies - which act as
   application-level intermediaries for Web requests - and more
   specifically with caching proxies, which retain copies of previously
   requested resources in the hope of improving overall quality of
   service by serving the content locally. Commonly used terminology in
   this memo can be found in the "Internet Web Replication and Caching
   Taxonomy"[2].

   No individual or organization has complete knowledge of the known
   problems in Web caching, and the editors are grateful to the
   contributors to this document.

1.1 Problem Template

   A common problem template is used within the following sections. We
   gratefully acknowledge RFC2525 [1] which helped define an initial
   format for this known problems list. The template format is
   summarized in the following table and described in more detail below.

      Name: short, descriptive name of the problem (3-5 words)
      Classification: classifies the problem: performance, security, etc
      Description: describes the problem succinctly
      Significance: magnitude of problem, environments where it exists
      Implications: the impact of the problem on systems and networks
      See Also: a reference to a related known problem
      Indications: states how to detect the presence of this problem

          Solution(s):   describe the solution(s) to this problem, if any
          Workaround: practical workaround for the problem
          References: information about the problem or solution
          Contact: contact name and email address for this section

   Name
      A short, descriptive, name (3-5 words) name associated with the
      problem.

   Classification
      Problems are grouped into categories of similar problems for ease
      of reading of this memo. Choose the category that best describes
      the problem. The suggested categories include three general
      categories and several more specific categories.

      * Architecture: the fundamental design is incomplete, or
         incorrect

      * Specification: the spec is ambiguous, incomplete, or incorrect.

      * Implementation: the implementation of the spec is incorrect.

      * Performance: perceived page response at the client is
         excessive; network bandwidth consumption is excessive; demand
         on origin or proxy servers exceed reasonable bounds.

      * Administration: care and feeding of caches is, or causes, a
         problem.

      * Security: privacy, integrity, or authentication concerns.

   Description
      A definition of the problem, succinct but including necessary
      background information.

   Significance (High, Medium, Low)
      May include a brief summary of the environments for which the
      problem is significant.

   Implications
      Why the problem is viewed as a problem. What inappropriate
      behavior results from it? This section should substantiate the
      magnitude of any problem indicated with High significance.

   See Also
      Optional. List of other known problems that are related to this
      one.

Indications
   How to detect the presence of the problem. This may include
   references to one or more substantiating documents that
   demonstrate the problem. This should include the network
   configuration that led to the problem such that it can be
   reproduced. Problems that are not reproducible will not appear in
   this memo.

Solution(s)
   Solutions that permanently fix the problem, if such are known. For
   example, what version of the software does not exhibit the
   problem? Indicate if the solution is accepted by the community,
   one of several solutions pending agreement, or open possibly with
   experimental solutions.

Workaround
   Practical workaround if no solution is available or usable. The
   workaround should have sufficient detail for someone experiencing
   the problem to get around it.

References
   References to related information in technical publications or on
   the web. Where can someone interested in learning more go to find
   out more about this problem, its solution, or workarounds?

Contact
   Contact name and email address of the person who supplied the
   information for this section. The editors are listed as contacts
   for anonymous submissions.

2. Known Problems

   The remaining sections of this document present the currently
   documented known problems. The problems are ordered by
   classification and significance. Issues with protocol specification
   or architecture are first, followed by implementation issues. Issues
   of high significance are first, followed by lower significance.

   Some of the problems initially identified in the previous versions of
   this document have been moved to Appendix A since they discuss issues
   where resolution primarily involves education rather than protocol
   work.

   A full list of the problems is available in the table of contents.

2.1 Known Specification Problems

2.1.1 Vary header is underspecified and/or misleading

   Name
      The "Vary" header is underspecified and/or misleading

   Classification
      Specification

   Description
      The Vary header in HTTP/1.1 was designed to allow a caching proxy
      to safely cache responses even if the server's choice of variants
      is not entirely understood. As RFC 2616 says:

         The Vary header field can be used to express the parameters the
         server uses to select a representation that is subject to
         server-driven negotiation.

      One might expect that this mechanism is useful in general for
      extensions that change the response message based on some aspects
      of the request. However, that is not true.

      During the design of the HTTP delta encoding specification[9] it
      was realized that an HTTP/1.1 proxy that does not understand delta
      encoding might cache a delta-encoded response and then later
      deliver it to a non-delta-capable client, unless the extension
      included some mechanism to prevent this. Initially, it was
      thought that Vary would suffice, but the following scenario proves
      this wrong.

      NOTE: It is likely that other scenarios exhibiting the same basic
      problem with "Vary" could be devised, without reference to delta
      encoding. This is simply a concrete scenario used to explain the
      problem.

      A complete description of the IM and A-IM headers may be found in
      the "Delta encoding in HTTP" specification. For the purpose of
      this problem description, the relevant details are:

      1. The concept of an "instance manipulation" is introduced. In
         some ways, this is similar to a content-coding, but there are
         differences. One example of an instance manipulation name is
         "vcdiff".

      2. A client signals its willingness to accept one or more
         instance-manipulations using the A-IM header.

   3. A server indicates which instance-manipulations are used to
      encode the body of a response using the IM header.

   4. Existing implementations will ignore the A-IM and IM headers,
      following the usual HTTP rules for handling unknown headers.

   5. Responses encoded with an instance-manipulation are sent using
      the (proposed) 226 status code, "IM Used".

   6. In response to a conditional request that carries an IM header,
      if the request-URI has been modified then a server may transmit
      a compact encoding of the modifications using a delta-encoding
      instead of a status-200 response. The encoded response cannot
      be understood by an implementation that does not support delta
      encodings.

   This summary omits many details.

   Suppose client A sends this request via proxy P:

      GET http://example.com/foo.html HTTP/1.1
      Host: example.com
      If-None-Match: "abc"
      A-IM: vcdiff

   and the origin server returns, via P, this response:

      HTTP/1.1 226 IM Used
      Etag: "def"
      Date: Wed, 19 Apr 2000 18:46:13 GMT
      IM: vcdiff
      Cache-Control: max-age-60
      Vary: A-IM, If-None-Match

   the body of which is a delta-encoded response (it encodes the
   difference between the Etag "abc" instance of foo.html, and the
   "def" instance). Assume that P stores this response in its cache,
   and that P does not understand the vcdiff encoding.

   Later, client B, also ignorant of delta-encoding, sends this
   request via P:

      GET http://example.com/foo.html HTTP/1.1
      Host: example.com

   What can P do now? According to the specification for the Vary
   header in RFC2616,

The Vary field value indicates the set of request-header fields
that fully determines, while the response is fresh, whether a
cache is permitted to use the response to reply to a subsequent
request without revalidation.

Implicitly, however, the cache would be allowed to use the stored
response in response to client B WITH "revalidation". This is the
potential bug.

An obvious implementation of the proxy would send this request to
test whether its cache entry is fresh (i.e., to revalidate the
entry):

    GET /foo.html HTTP/1.1
    Host: example.com
    If-None-Match: "def"

That is, the proxy simply forwards the new request, after doing
the usual transformation on the URL and tacking on the "obvious"
If-None-Match header.

If the origin server's Etag for the current instance is still
"def", it would naturally respond:

    HTTP/1.1 304 Not Modified
    Etag: "def"
    Date: Wed, 19 Apr 2000 18:46:14 GMT

thus telling the proxy P that it can use its stored response. But
this cache response actually involves a delta-encoding that would
not be sensible to client B, signaled by a header field that would
be ignored by B, and so the client displays garbage.

The problem here is that the original request (from client A)
generated a response that is not sensible to client B, not merely
one that is not "the appropriate representation" (as the result of
server-driven negotiation).

One might argue that the proxy P shouldn't be storing status-226
responses in the first place. True in theory, perhaps, but
unfortunately RFC2616, section 13.4, says:

    A response received with any [status code other than 200, 203,
    206, 300, 301 or 410] MUST NOT be returned in a reply to a
    subsequent request unless there are cache-control directives or
    another header(s) that explicitly allow it. For example, these

include the following: an Expires header (section 14.21); a
"max-age", "s-maxage", "must-revalidate", "proxy-revalidate",
"public" or "private" cache-control directive (section 14.9).

In other words, the specification allows caching of responses with
yet-to-be-defined status codes if the response carries a plausible
Cache-Control directive. So unless we ban servers implementing
this kind of extension from using these Cache-Control directives
at all, the Vary header just won't work.

Significance
    Medium

Implications
    Certain plausible extensions to the HTTP/1.1 protocol might not
    interoperate correctly with older HTTP/1.1 caches, if the
    extensions depend on an interpretation of Vary that is not the
    same as is used by the cache implementer.

    This would have the effect either of causing hard-to-debug cache
    transparency failures, or of discouraging the deployment of such
    extensions, or of encouraging the implementers of such extensions
    to disable caching entirely.

Indications
    The problem is visible when hand-simulating plausible message
    exchanges, especially when using the proposed delta encoding
    extension. It probably has not been visible in practice yet.

Solution(s)

    1. Section 13.4 of the HTTP/1.1 specification should probably be
       changed to prohibit caching of responses with status codes that
       the cache doesn't understand, whether or not they include
       Expires headers and the like. (It might require some care to
       define what "understands" means, leaving room for future
       extensions with new status codes.) The behavior in this case
       needs to be defined as equivalent to "Cache-Control: no-store"
       rather than "no-cache", since the latter allows revalidation.

       Possibly the specification of Vary should require that it be
       treated as "Cache-Control: no-store" whenever the status code
       is unknown - that should solve the problem in the scenario
       given here.

2. Designers of HTTP/1.1 extensions should consider using
   mechanisms other than Vary to prevent false caching.

   It is not clear whether the Vary mechanism is widely
   implemented in caches; if not, this favors solution #1.

Workaround
   A cache could treat the presence of a Vary header in a response as
   an implicit "Cache-control: no-store", except for "known" status
   codes, even though this is not required by RFC 2616. This would
   avoid any transparency failures. "Known status codes" for basic
   HTTP/1.1 caches probably include: 200, 203, 206, 300, 301, 410
   (although this list should be re-evaluated in light of the problem
   discussed here).

References
   See [9] for the specification of the delta encoding extension, as
   well as for an example of the use of a Cache-Control extension
   instead of "Vary."

Contact
   Jeff Mogul <mogul@pa.dec.com>

2.1.2 Client Chaining Loses Valuable Length Meta-Data

Name
   Client Chaining Loses Valuable Length Meta-Data

Classification
   Performance

Description
   HTTP/1.1[3] implementations are prohibited from sending Content-
   Length headers with any message whose body has been Transfer-
   Encoded. Because 1.0 clients cannot accept chunked Transfer-
   Encodings, receiving 1.1 implementations must forward the body to
   1.0 clients must do so without the benefit of information that was
   discarded earlier in the chain.

Significance
   Low

Implications
   Lacking either a chunked transfer encoding or Content-Length
   indication creates negative performance implications for how the
   proxy must forward the message body.

In the case of response bodies, the server may either forward the
response while closing the connection to indicate the end of the
response or must utilize store and forward semantics to buffer the
entire response in order to calculate a Content-Length. The
former option defeats the performance benefits of persistent
connections in HTTP/1.1 (and their Keep-Alive cousin in HTTP/1.0)
as well as creating some ambiguously lengthed responses. The
latter store and forward option may not even be feasible given the
size of the resource and it will always introduce increased
latency.

Request bodies must undertake the store and forward process as 1.0
request bodies must be delimited by Content-Length headers. As
with response bodies this may place unacceptable resource
constraints on the proxy and the request may not be able to be
satisfied.

Indications
   The lack of HTTP/1.0 style persistent connections between 1.0
   clients and 1.1 proxies, only when accessing 1.1 servers, is a
   strong indication of this problem.

Solution(s)
   An HTTP specification clarification that would allow origin known
   identity document Content-Lengths to be carried end to end would
   alleviate this issue.

Workaround
   None.

Contact
   Patrick McManus <mcmanus@AppliedTheory.com>

2.2 Known Architectural Problems

2.2.1 Interception proxies break client cache directives

   Name
      Interception proxies break client cache directives

   Classification
      Architecture

   Description
      HTTP[3] is designed for the user agent to be aware if it is
      connected to an origin server or to a proxy. User agents
      believing they are transacting with an origin server but which are

really in a connection with an interception proxy may fail to send
critical cache-control information they would have otherwise
included in their request.

Significance
   High

Implications
   Clients may receive data that is not synchronized with the origin
   even when they request an end to end refresh, because of the lack
   of inclusion of either a "Cache-control: no-cache" or "must-
   revalidate" header. These headers have no impact on origin server
   behavior so may not be included by the browser if it believes it
   is connected to that resource. Other related data implications
   are possible as well. For instance, data security may be
   compromised by the lack of inclusion of "private" or "no-store"
   clauses of the Cache-control header under similar conditions.

Indications
   Easily detected by placing fresh (un-expired) content on a caching
   proxy while changing the authoritative copy, then requesting an
   end-to-end reload of the data through a proxy in both interception
   and explicit modes.

Solution(s)
   Eliminate the need for interception proxies and IP spoofing, which
   will return correct context awareness to the client.

Workaround
   Include relevant Cache-Control directives in every request at the
   cost of increased bandwidth and CPU requirements.

Contact
   Patrick McManus <mcmanus@AppliedTheory.com>

2.2.2 Interception proxies prevent introduction of new HTTP methods

Name
   Interception proxies prevent introduction of new HTTP methods

Classification
   Architecture

Description
   A proxy that receives a request with a method unknown to it is
   required to generate an HTTP 501 Error as a response. HTTP
   methods are designed to be extensible so there may be applications
   deployed with initial support just for the user agent and origin

server. An interception proxy that hijacks requests which include
new methods destined for servers that have implemented those
methods creates a de-facto firewall where none may be intended.

Significance
Medium within interception proxy environments.

Implications
Renders new compliant applications useless unless modifications
are made to proxy software. Because new methods are not required
to be globally standardized it is impossible to keep up to date in
the general case.

Solution(s)
Eliminate the need for interception proxies. A client receiving a
501 in a traditional HTTP environment may either choose to repeat
the request to the origin server directly, or perhaps be
configured to use a different proxy.

Workaround
Level 5 switches (sometimes called Level 7 or application layer
switches) can be used to keep HTTP traffic with unknown methods
out of the proxy. However, these devices have heavy buffering
responsibilities, still require TCP sequence number spoofing, and
do not interact well with persistent connections.

The HTTP/1.1 specification allows a proxy to switch over to tunnel
mode when it receives a request with a method or HTTP version it
does not understand how to handle.

Contact
Patrick McManus <mcmanus@AppliedTheory.com>
Henrik Nordstrom <hno@hem.passagen.se> (HTTP/1.1 clarification)

2.2.3 Interception proxies break IP address-based authentication

Name
Interception proxies break IP address-based authentication

Classification
Architecture

Description
Some web servers are not open for public access, but restrict
themselves to accept only requests from certain IP address ranges
for security reasons. Interception proxies alter the source
(client) IP addresses to that of the proxy itself, without the

knowledge of the client/user. This breaks such authentication
mechanisms and prohibits otherwise allowed clients access to the
servers.

   Significance
      Medium

   Implications
      Creates end user confusion and frustration.

   Indications
      Users may start to see refused connections to servers after
      interception proxies are deployed.

   Solution(s)
      Use user-based authentication instead of (IP) address-based
      authentication.

   Workaround
      Using IP filters at the intercepting device (L4 switch) and bypass
      all requests to such servers concerned.

   Contact
      Keith K. Chau <keithc@unitechnetworks.com>

2.2.4 Caching proxy peer selection in heterogeneous networks

   Name
      Caching proxy peer selection in heterogeneous networks

   Classification
      Architecture

   Description
      ICP[4] based caching proxy peer selection in networks with large
      variance in latency and bandwidth between peers can lead to non-
      optimal peer selection. For example take Proxy C with two
      siblings, Sib1 and Sib2, and the following network topology
      (summarized).

      * Cache C's link to Sib1, 2 Mbit/sec with 300 msec latency

      * Cache C's link to Sib2, 64 Kbit/sec with 10 msec latency.

      ICP[4] does not work well in this context. If a user submits a
      request to Proxy C for page P that results in a miss, C will send
      an ICP request to Sib1 and Sib2. Assume both siblings have the

      requested object P. The ICP_HIT reply will always come from Sib2
      before Sib1. However, it is clear that the retrieval of large
      objects will be faster from Sib1, rather than Sib2.

      The problem is more complex because Sib1 and Sib2 can't have a
      100% hit ratio. With a hit rate of 10%, it is more efficient to
      use Sib1 with resources larger than 48K. The best choice depends
      on at least the hit rate and link characteristics; maybe other
      parameters as well.

   Significance
      Medium

   Implications
      By using the first peer to respond, peer selection algorithms are
      not optimizing retrieval latency to end users. Furthermore they
      are causing more work for the high-latency peer since it must
      respond to such requests but will never be chosen to serve content
      if the lower latency peer has a copy.

   Indications
      Inherent in design of ICP v1, ICP v2, and any cache mesh protocol
      that selects peers based upon first response.

      This problem is not exhibited by cache digest or other protocols
      which (attempt to) maintain knowledge of peer contents and only
      hit peers that are believed to have a copy of the requested page.

   Solution(s)
      This problem is architectural with the peer selection protocols.

   Workaround
      Cache mesh design when using such a protocol should be done in
      such a way that there is not a high latency variance among peers.
      In the example presented in the above description the high latency
      high bandwidth peer could be used as a parent, but should not be
      used as a sibling.

   Contact
      Ivan Lovric <ivan.lovric@cnet.francetelecom.fr>
      John Dilley <jad@akamai.com>

2.2.5 ICP Performance

   Name
      ICP performance

   Classification
      Architecture(ICP), Performance

   Description
      ICP[4] exhibits $O(n^2)$ scaling properties, where n is the number
      of participating peer proxies. This can lead ICP traffic to
      dominate HTTP traffic within a network.

   Significance
      Medium

   Implications
      If a proxy has many ICP peers the bandwidth demand of ICP can be
      excessive. System managers must carefully regulate ICP peering.
      ICP also leads proxies to become homogeneous in what they serve;
      if your proxy does not have a document it is unlikely your peers
      will have it either. Therefore, ICP traffic requests are largely
      unable to locate a local copy of an object (see [6]).

   Indications
      Inherent in design of ICP v1, ICP v2.

   Solution(s)
      This problem is architectural - protocol redesign or replacement
      is required to solve it if ICP is to continue to be used.

   Workaround
      Implementation workarounds exist, for example to turn off use of
      ICP, to carefully regulate peering, or to use another mechanism if
      available, such as cache digests. A cache digest protocol shares
      a summary of cache contents using a Bloom Filter technique. This
      allows a cache to estimate whether a peer has a document. Filters
      are updated regularly but are not always up-to-date so cannot help
      when a spike in popularity occurs. They also increase traffic but
      not as much as ICP.

      Proxy clustering protocols organize proxies into a mesh provide
      another alternative solution. There is ongoing research on this
      topic.

   Contact
      John Dilley <jad@akamai.com>

2.2.6 Caching proxy meshes can break HTTP serialization of content

   Name
      Caching proxy meshes can break HTTP serialization of content

   Classification
      Architecture (HTTP protocol)

   Description
      A caching proxy mesh where a request may travel different paths,
      depending on the state of the mesh and associated caches, can
      break HTTP content serialization, possibly causing the end user to
      receive older content than seen on an earlier request, where the
      request traversed another path in the mesh.

   Significance
      Medium

   Implications
      Can cause end user confusion. May in some situations (sibling
      cache hit, object has changed state from cacheable to uncacheable)
      be close to impossible to get the caches properly updated with the
      new content.

   Indications
      Older content is unexpectedly returned from a caching proxy mesh
      after some time.

   Solutions(s)
      Work with caching proxy vendors and researchers to find a suitable
      protocol for maintaining proxy relations and object state in a
      mesh.

   Workaround
      When designing a hierarchy/mesh, make sure that for each end-
      user/URL combination there is only one single path in the mesh
      during normal operation.

   Contact
      Henrik Nordstrom <hno@hem.passagen.se>

2.3 Known Implementation Problems

2.3.1 User agent/proxy failover

   Name
      User agent/proxy failover

   Classification
      Implementation

   Description
      Failover between proxies at the user agent (using a proxy.pac[8]
      file) is erratic and no standard behavior is defined.
      Additionally, behavior is hard-coded into the browser, so that
      proxy administrators cannot use failover at the user agent
      effectively.

   Significance
      Medium

   Implications
      Architects are forced to implement failover at the proxy itself,
      when it may be more appropriate and economical to do it within the
      user agent.

   Indications
      If a browser detects that its primary proxy is down, it will wait
      n minutes before trying the next one it is configured to use. It
      will then wait y minutes before asking the user if they'd like to
      try the original proxy again. This is very confusing for end
      users.

   Solution(s)
      Work with browser vendors to establish standard extensions to
      JavaScript proxy.pac libraries that will allow configuration of
      these timeouts.

   Workaround
      User education; redundancy at the proxy level.

   Contact
      Mark Nottingham <mnot@mnot.net>

2.3.2 Some servers send bad Content-Length headers for files that
      contain CR

   Name
      Some servers send bad Content-Length headers for files that
      contain CR

   Classification
      Implementation

   Description
      Certain web servers send a Content-length value that is larger
      than number of bytes in the HTTP message body. This happens when
      the server strips off CR characters from text files with lines
      terminated with CRLF as the file is written to the client. The
      server probably uses the stat() system call to get the file size
      for the Content-Length header. Servers that exhibit this behavior
      include the GN Web server (version 2.14 at least).

   Significance
      Low. Surveys indicate only a small number of sites run faulty
      servers.

   Implications
      In this case, an HTTP client (e.g., user agent or proxy) may
      believe it received a partial response. HTTP/1.1 [3] advises that
      caches MAY store partial responses.

   Indications
      Count the number of bytes in the message body and compare to the
      Content-length value. If they differ the server exhibits this
      problem.

   Solutions
      Upgrade or replace the buggy server.

   Workaround
      Some browsers and proxies use one TCP connection per object and
      ignore the Content-Length. The document end of file is identified
      by the close of the TCP socket.

   Contact
      Duane Wessels <wessels@measurement-factory.com>

3. Security Considerations

   This memo does not raise security considerations in itself. See the
   individual submissions for details of security concerns and issues.

References

   [1] Paxson, V., Allman, M., Dawson, S., Fenner, W., Griner, J.,
        Heavens, I., Lahey, K., Semke, J. and B. Volz, "Known TCP
        Implementation Problems", RFC 2525, March 1999.

   [2] Cooper, I., Melve, I. and G. Tomlinson, "Internet Web
        Replication and Caching Taxonomy", RFC 3040, January 2001.

   [3] 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.

   [4] Wessels, D. and K. Claffy, "Internet Cache Protocol (ICP),
        Version 2", RFC 2186, September 1997.

   [5] Davison, B., "Web Traffic Logs: An Imperfect Resource for
        Evaluation", in Proceedings of the Ninth Annual Conference of
        the Internet Society (INET'99), July 1999.

   [6] Melve, I., "Relation Analysis, Cache Meshes", in Proceedings of
        the 3rd International WWW Caching Workshop, June 1998,
        <http://wwwcache.ja.net/events/workshop/29/magicnumber.html>.

   [7] Krishnamurthy, B. and M. Arlett, "PRO-COW: Protocol Compliance
        on the Web", AT&T Labs Technical Report #990803-05-TM, August
        1999, <http://www.research.att.com/˜bala/papers/procow-1.ps.gz>.

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

   [9] Mogul, J., Krishnamurthy, B., Douglis, F., Feldmann, A., Goland,
        Y., van Hoff, A. and D. Hellerstein, "HTTP Delta in HTTP", Work
        in Progress.

Authors' Addresses

    Ian Cooper
    Equinix, Inc.
    2450 Bayshore Parkway
    Mountain View, CA 94043
    USA

    Phone: +1 650 316 6065
    EMail: icooper@equinix.com

    John Dilley
    Akamai Technologies, Inc.
    1400 Fashion Island Blvd
    Suite 703
    San Mateo, CA 94404
    USA

    Phone: +1 650 627 5244
    EMail: jad@akamai.com

Appendix A. Archived Known Problems

   The following sub-sections are an archive of problems identified in
   the initial production of this memo. These are typically problems
   requiring further work/research, or user education. They are
   included here for reference purposes only.

A.1 Architectural

A.1.1 Cannot specify multiple URIs for replicated resources

   Name
      Cannot specify multiple URIs for replicated resources

   Classification
      Architecture

   Description
      There is no way to specify that multiple URIs may be used for a
      single resource, one for each replica of the resource. Similarly,
      there is no way to say that some set of proxies (each identified
      by a URI) may be used to resolve a URI.

   Significance
      Medium

   Implications
      Forces users to understand the replication model and mechanism.
      Makes it difficult to create a replication framework without
      protocol support for replication and naming.

   Indications
      Inherent in HTTP/1.0, HTTP/1.1.

   Solution(s)
      Architectural - protocol design is necessary.

   Workaround
      Replication mechanisms force users to locate a replica or mirror
      site for replicated content.

   Contact
      Daniel LaLiberte <liberte@w3.org>

A.1.2 Replica distance is unknown

   Name
      Replica distance is unknown

   Classification
      Architecture

   Description
      There is no recommended way to find out which of several servers
      or proxies is closer either to the requesting client or to another
      machine, either geographically or in the network topology.

   Significance
      Medium

   Implications
      Clients must guess which replica is closer to them when requesting
      a copy of a document that may be served from multiple locations.
      Users must know the set of servers that can serve a particular
      object. This in general is hard to determine and maintain. Users
      must understand network topology in order to choose the closest
      copy. Note that the closest copy is not always the one that will
      result in quickest service. A nearby but heavily loaded server
      may be slower than a more distant but lightly loaded server.

   Indications
      Inherent in HTTP/1.0, HTTP/1.1.

   Solution(s)
      Architectural - protocol work is necessary. This is a specific
      instance of a general problem in widely distributed systems. A
      general solution is unlikely, however a specific solution in the
      web context is possible.

   Workaround
      Servers can (many do) provide location hints in a replica
      selection web page. Users choose one based upon their location.
      Users can learn which replica server gives them best performance.
      Note that the closest replica geographically is not necessarily
      the closest in terms of network topology. Expecting users to
      understand network topology is unreasonable.

   Contact
      Daniel LaLiberte <liberte@w3.org>

A.1.3 Proxy resource location

   Name
      Proxy resource location

   Classification
      Architecture

   Description
      There is no way for a client or server (including another proxy)
      to inform a proxy of an alternate address (perhaps including the
      proxy to use to reach that address) to use to fetch a resource.
      If the client does not trust where the redirected resource came
      from, it may need to validate it or validate where it came from.

   Significance
      Medium

   Implications
      Proxies have no systematic way to locate resources within other
      proxies or origin servers. This makes it more difficult to share
      information among proxies. Information sharing would improve
      global efficiency.

   Indications
      Inherent in HTTP/1.0, HTTP/1.1.

   Solution(s)
      Architectural - protocol design is necessary.

   Workaround
      Certain proxies share location hints in the form of summary
      digests of their contents (e.g., Squid). Certain proxy protocols
      enable a proxy query another for its contents (e.g., ICP). (See
      however "ICP Performance" issue (Section 2.2.5).)

   Contact
      Daniel LaLiberte <liberte@w3.org>

A.2 Implementation

A.2.1 Use of Cache-Control headers

   Name
      Use of Cache-Control headers

   Classification
      Implementation

Description
    Many (if not most) implementations incorrectly interpret Cache-
    Control response headers.

Significance
    High

Implications
    Cache-Control headers will be spurned by end users if there are
    conflicting or non-standard implementations.

Indications
    -

Solution(s)
    Work with vendors and others to assure proper application

Workaround
    None.

Contact
    Mark Nottingham <mnot@mnot.net>

A.2.2 Lack of HTTP/1.1 compliance for caching proxies

Name
    Lack of HTTP/1.1 compliance for caching proxies

Classification
    Implementation

Description
    Although performance benchmarking of caches is starting to be
    explored, protocol compliance is just as important.

Significance
    High

Implications
    Caching proxy vendors implement their interpretation of the
    specification; because the specification is very large, sometimes
    vague and ambiguous, this can lead to inconsistent behavior
    between caching proxies.

    Caching proxies need to comply to the specification (or the
    specification needs to change).

   Indications
      There is no currently known compliance test being used.

      There is work underway to quantify how closely servers comply with
      the current specification. A joint technical report between AT&T
      and HP Labs [7] describes the compliance testing. This report
      examines how well each of a set of top traffic-producing sites
      support certain HTTP/1.1 features.

      The Measurement Factory (formerly IRCache) is working to develop
      protocol compliance testing software. Running such a conformance
      test suite against caching proxy products would measure compliance
      and ultimately would help assure they comply to the specification.

   Solution(s)
      Testing should commence and be reported in an open industry forum.
      Proxy implementations should conform to the specification.

   Workaround
      There is no workaround for non-compliance.

   Contact
      Mark Nottingham <mnot@mnot.net>
      Duane Wessels <wessels@measurement-factory.com>

A.2.3 ETag support

   Name
      ETag support

   Classification
      Implementation

   Description
      Available caching proxies appear not to support ETag (strong)
      validation.

   Significance
      Medium

   Implications
      Last-Modified/If-Modified-Since validation is inappropriate for
      many requirements, both because of its weakness and its use of
      dates. Lack of a usable, strong coherency protocol leads
      developers and end users not to trust caches.

   Indications
      -

Solution(s)
   Work with vendors to implement ETags; work for better validation
   protocols.

Workaround
   Use Last-Modified/If-Modified-Since validation.

Contact
   Mark Nottingham <mnot@mnot.net>

A.2.4 Servers and content should be optimized for caching

Name
   Servers and content should be optimized for caching

Classification
   Implementation (Performance)

Description
   Many web servers and much web content could be implemented to be
   more conducive to caching, reducing bandwidth demand and page load
   delay.

Significance
   Medium

Implications
   By making poor use of caches, origin servers encourage longer load
   times, greater load on caching proxies, and increased network
   demand.

Indications
   The problem is most apparent for pages that have low or zero
   expires time, yet do not change.

Solution(s)
   -

Workaround
   Servers could start using unique object identifiers for write-only
   content: if an object changes it gets a new name, otherwise it is
   considered to be immutable and therefore have an infinite expire
   age. Certain hosting providers do this already.

Contact
   Peter Danzig

A.3 Administration

A.3.1 Lack of fine-grained, standardized hierarchy controls

   Name
      Lack of fine-grained, standardized hierarchy controls

   Classification
      Administration

   Description
      There is no standard for instructing a proxy as to how it should
      resolve the parent to fetch a given object from. Implementations
      therefore vary greatly, and it can be difficult to make them
      interoperate correctly in a complex environment.

   Significance
      Medium

   Implications
      Complications in deployment of caches in a complex network
      (especially corporate networks)

   Indications
      Inability of some proxies to be configured to direct traffic based
      on domain name, reverse lookup IP address, raw IP address, in
      normal operation and in failover mode. Inability in some proxies
      to set a preferred parent / backup parent configuration.

   Solution(s)
      -

   Workaround
      Work with vendors to establish an acceptable configuration within
      the limits of their product; standardize on one product.

   Contact
      Mark Nottingham <mnot@mnot.net>

A.3.2 Proxy/Server exhaustive log format standard for analysis

   Name
      Proxy/Server exhaustive log format standard for analysis

   Classification
      Administration

   Description
      Most proxy or origin server logs used for characterization or
      evaluation do not provide sufficient detail to determine
      cacheability of responses.

   Significance
      Low (for operationality; high significance for research efforts)

   Implications
      Characterizations and simulations are based on non-representative
      workloads.

   See Also
      W3C Web Characterization Activity, since they are also concerned
      with collecting high quality logs and building characterizations
      from them.

   Indications
      -

   Solution(s)
      To properly clean and to accurately determine cacheability of
      responses, a complete log is required (including all request
      headers as well as all response headers such as "User-agent" [for
      removal of spiders] and "Expires", "max-age", "Set-cookie", "no-
      cache", etc.)

   Workaround
      -

   References
      See "Web Traffic Logs: An Imperfect Resource for Evaluation"[5]
      for some discussion of this.

   Contact
      Brian D. Davison <davison@acm.org>
      Terence Kelly <tpkelly@eecs.umich.edu>

A.3.3 Trace log timestamps

   Name
      Trace log timestamps

   Classification
      Administration

Description
    Some proxies/servers log requests without sufficient timing
    detail. Millisecond resolution is often too small to preserve
    request ordering and either the servers should record request
    reception time in addition to completion time, or elapsed time
    plus either one.

Significance
    Low (for operationality; medium significance for research efforts)

Implications
    Characterization and simulation fidelity is improved with accurate
    timing and ordering information. Since logs are generally written
    in order of request completion, these logs cannot be re-played
    without knowing request generation times and reordering
    accordingly.

See Also
    -

Indications
    Timestamps can be identical for multiple entries (when only
    millisecond resolution is used). Request orderings can be jumbled
    when clients open additional connections for embedded objects
    while still receiving the container object.

Solution(s)
    Since request completion time is common (e.g., Squid), recommend
    continuing to use it (with microsecond resolution if possible)
    plus recording elapsed time since request reception.

Workaround
    -

References
    See "Web Traffic Logs: An Imperfect Resource for Evaluation"[5]
    for some discussion of this.

Contact
    Brian D. Davison <davison@acm.org>

A.3.4 Exchange format for log summaries

Name
    Exchange format for log summaries

Classification
    Administration/Analysis?

Description
    Although we have (more or less) a standard log file format for
    proxies (plain vanilla Common Logfile and Squid), there isn't a
    commonly accepted format for summaries of those log files.
    Summaries could be generated by the cache itself, or by post-
    processing existing log file formats such as Squid's.

Significance
    High, since it means that each log file summarizing/analysis tool
    is essentially reinventing the wheel (un-necessary repetition of
    code), and the cost of processing a large number of large log
    files through a variety of analysis tools is (again for no good
    reason) excessive.

Implications
    In order to perform a meaningful analysis (e.g., to measure
    performance in relation to loading/configuration over time) the
    access logs from multiple busy caches, it's often necessary to run
    first one tool then another, each against the entire log file (or
    a significantly large subset of the log). With log files running
    into hundreds of MB even after compression (for a cache dealing
    with millions of transactions per day) this is a non-trivial task.

See Also
    IP packet/header sniffing - it may be that individual transactions
    are at a level of granularity which simply isn't sensible to be
    attempting on extremely busy caches. There may also be legal
    implications in some countries, e.g., if this analysis identifies
    individuals.

Indications
    Disks/memory full(!) Stats (using multiple programs) take too long
    to run. Stats crunching must be distributed out to multiple
    machines because of its high computational cost.

Solution(s)
    Have the proxy produce a standardized summary of its activity
    either automatically or via an external (e.g., third party) tool,
    in a commonly agreed format. The format could be something like
    XML or the Extended Common Logfile, but the format and contents
    are subjects for discussion. Ideally this approach would permit
    individual cache server products to supply subsets of the possible
    summary info, since it may not be feasible for all servers to
    provide all of the information which people would like to see.

   Workaround
      Devise a private summary format for your own personal use - but
      this complicates or even precludes the exchange of summary info
      with other interested parties.

   References
      See the web pages for the commonly used cache stats analysis
      programs, e.g., Calamaris, squidtimes, squidclients, etc.

   Contact
      Martin Hamilton <martin@wwwcache.ja.net>

Full Copyright Statement

Acknowledgement