Network Working Group   A. Barbir
Internet-Draft   Nortel Networks
Expires: August 22, 2002   B. Cain
                                                  Cereva Networks
                                               F. Douglis
                                                        AT&T Labs
                                               M. Green
                                                        CacheFlow
                                               M. Hofmann
                                                            Lucent
                                               R. Nair
                                               D. Potter
                                                            Cisco
                                               O. Spatscheck
                                                        AT&T Labs

                                               February 22, 2002

Known CN Request-Routing Mechanisms
draft-ietf-cdi-known-request-routing-00.txt

Status of this Memo

Copyright Notice

Abstract

   The work presents a summary of Request-Routing techniques that are
   used to direct client requests to surrogates based on various
   policies and a possible set of metrics. In this memo the term
   Request-Routing represents techniques that are commonly called
   content routing or content redirection. In principle,
   Request-Routing techniques can be classified under: DNS
   Request-Routing, Transport-layer Request-Routing, and
   Application-layer Request-Routing.

Table of Content

1.   Introduction

   The document provides a summary of current known techniques that
   could be used to direct client requests to surrogates based on
   various policies and a possible set of metrics. The task of
   directing clients' requests to surrogates is also called
   Request-Routing, Content Routing or Content Redirection.

   Request-Routing techniques are commonly used in Content Networks
   (also known as Content Delivery Networks)[5]. Content Networks
   include network infrastructure that exists in layers 4 through 7.
   Content Networks deal with the routing and forwarding of requests
   and responses for content. Content Networks rely on layer 7
   protocols such as HTTP [4] OR RTP [8] for transport.

   Request-Routing techniques are generally used to direct client
   requests for objects to a surrogate or a set of surrogates that
   could best serve that content. Request-Routing mechanisms could
   be used to direct client requests to surrogates that are within a
   Content Network (CN) [5].

   Request-Routing techniques are used as a vehicle to extend the
   reach and scale of Content Delivery Networks. There exist multiple
   Request-Routing mechanisms. At a high-level, these may be classified
   under: DNS Request-Routing, transport-layer Request-Routing, and
   application-layer Request-Routing.

   A request routing system uses a set of metrics in an attempt to
   direct users to surrogate that can best serve the request. For
   example, the choice of the surrogate could be based on network
   proximity, bandwidth availability, surrogate load and availability
   of content. Appendix A provides a summary of metrics and
   measurement techniques that could be used in the selection of
   the best surrogate.

   The memo is organized as follows: Section 2 provides a summary of
   known DNS based Request-Routing techniques. Section 3 discusses
   transport-layer Request-Routing methods. In section 4
   application-layer Request-Routing mechanisms are explored. Section
   5 provides insight on combining the various methods that were
   discussed in the earlier sections in order to optimize the
   performance of the Request-Routing System. Appendix A provides
   a summary of possible metrics and measurements techniques that could
   be used by the Request-Routing system to choose a given surrogate.

2. DNS based Request-Routing Mechanisms

   DNS based Request-Routing techniques are common due to the ubiquity
   of DNS as a directory service. In DNS based Request-Routing

techniques, a specialized DNS server is inserted in the DNS

   resolution process. The server is capable of returning a different
   set of A, NS or CNAME records based on user defined policies,
   metrics, or a combination of both.

2.1 Single Reply

   In this approach, the DNS server is authoritative for the entire
   DNS domain or a sub domain. The DNS server returns the IP address
   of the best surrogate in an A record to the requesting DNS server.
   The IP address of the surrogate could also be a virtual IP(VIP)
   address of the best set of surrogates for requesting DNS server.

2.2 Multiple Replies

   In this approach, the Request-Routing DNS server returns multiple
   replies such as several A records for various surrogates. Common
   implementations of client site DNS server's cycles through the
   multiple replies in a Round-Robin fashion. The order in which the
   records are returned can be used to direct multiple clients using
   a single client site DNS server.

2.3 Multi-Level Resolution

   In this approach multiple Request-Routing DNS servers can be
   involved in a single DNS resolution. The rationale of utilizing
   multiple Request-Routing DNS servers in a single DNS resolution
   is to allow one to distribute more complex decisions from a
   single server to multiple, more specialized, Request-Routing DNS
   servers. The most common mechanisms used to insert multiple
   Request-Routing DNS servers in a single DNS resolution is the
   use of NS and CNAME records. An example would be the case where
   a higher level DNS server operates within a territory,
   directing the DNS lookup to a more specific DNS server within that
   territory to provide a more accurate resolution.

2.3.1 NS Redirection

   A DNS server can use NS records to redirect the authority of the
   next level domain to another Request-Routing DNS server. The,
   technique allows multiple DNS server to be involved in the name
   resolution process. For example, a client site DNS server
   resolving a.b.c.com would eventually request a resolution of
   a.b.c.com from the name server authoritative for c.com. The name
   server authoritative for this domain might be a Request-Routing
   NS server. In this case the Request-Routing DNS server can either
   return a set of A records or can redirect the resolution of the
   request a.b.c.com to the DNS server that is authoritative for
   .c.com using NS records.

   One drawback of using NS records is that the number of

Request-Routing   DNS servers are limited by the number of parts in

the DNS name. This problem results from DNS policy that causes a
client site DNS server to abandon a request if no additional parts
of the DNS name are resolved in an exchange with an authoritative
DNS server.

A second drawback is that the last DNS server can determine the
TTL of the entire resolution process. Basically, the last DNS
server can return in the authoritative section of its response its
own NS record. The client will use this cached NS record for
further request resolutions until it expires.

Another drawback is that some implementations of bind voluntarily
cause timeouts to simplify their implementation in cases in which a
NS level redirect points to a name server for which no valid A
record is returned or cached. This is especially a problem if the
domain of the name server does not match the domain currently
resolved, since in this case the A records, which might be passed
in the DNS response, are discarded for security reasons. Another
drawback is the added delay in resolving the request due to the
use of multiple DNS servers.

2.3.2 CNAME Redirection

In this scenario, the Request-Routing DNS server returns a CNAME
record to direct resolution to an entirely new domain. In
principle, the new domain might employ a new set of Request-Routing
DNS servers.

One disadvantage of this approach is the additional overhead of
resolving the new domain name. The main advantage of this approach
is that the number of Request-Routing DNS servers is independent
of the format of the domain name.

2.4 Anycast

Anycast [6] is an inter-network service that is applicable to
networking situations where a host, application, or user wishes to
locate a host which supports a particular service but, if several
servers support the service, does not particularly care which server
is used. In an anycast service, a host transmits a datagram to an
anycast address and the inter-network is responsible for providing
best effort delivery of the datagram to at least one, and
preferably only one, of the servers that accept datagrams for the
anycast address.

The motivation for anycast is that it considerably simplifies the
task of finding an appropriate server. For example, users, instead
of consulting a list of servers and choosing the closest one, could

simply type the name of the server and be connected to the nearest

one. By using anycast, DNS resolvers would no longer have to be
configured with the IP addresses of their servers, but rather could
send a query to a well-known DNS anycast address.

Furthermore, to combine measurement and redirection, the
Request-Routing DNS server can advertise an anycast address as its
IP address. The same address is used by multiple physical DNS
servers. In this scenario, the Request-Routing DNS server that is
the closest to the client site DNS server in terms of OSPF and
BGP routing will receive the packet containing the DNS resolution
request. The server can use this information to make a Request-
Routing decision. Drawbacks of this approach are listed below:

* The DNS server may not be the closest server in terms of
  routing to the client.

* Typically, routing protocols are not load sensitive. Hence,
  the closest server may not be the one with the least network
  latency.

* The server load is not considered during the Request-
  Routing process.

2.5 Object Encoding

Since only DNS names are visible during the DNS Request-Routing,
some solutions encode the object type, object hash, or similar
information into the DNS name. This might vary from a simple
division of objects based on object type (such as images.a.b.c.com
and streaming.a.b.c.com) to a sophisticated schema in which the
domain name contains a unique identifier (such as a hash) of the
object. The obvious advantage is that object information is
available at resolution time. The disadvantage is that the client
site DNS server has to perform multiple resolutions to retrieve a
single Web page, which might increase rather than decrease the
overall latency.

2.6 DNS Request-Routing Limitations

Some limitations of DNS based Request-Routing techniques are
described below:

1. DNS only allows resolution at the domain level. However, an
   ideal request resolution system should service requests
   per object level.

2. In DNS based Request-Routing systems servers may be required
   to return DNS entries with a short time-to-live (TTL) values.
   This may be needed in order to be able to react quickly in the

face of outages. This in return may increase the volume of
requests to DNS servers.

3. Some DNS implementations do not always adhere to DNS standards.
    For example, many DNS implementations do not honor the DNS TTL
    field.

4. DNS Request-Routing is based only on knowledge of the client
    DNS server, as client addresses are not relayed within DNS
    requests. This limits the ability of the Request-Routing system
    to determine a client's proximity to the surrogate.

5. DNS servers can request and allow recursive resolution of DNS
    names. For recursive resolution of requests, the Request-
    Routing DNS server will not be exposed to the IP address of the
    client's site DNS server. In this case, the Request-Routing DNS
    server will be exposed to the address of the DNS server that is
    recursively requesting the information on behalf of the client's
    site DNS server. For example, imgs.company.com might be resolved
    by a CN, but the request for the resolution might come from
    dns1.company.com as a result of the recursion.

6. Users that share a single client site DNS server will be
    redirected to the same set of IP addresses during the TTL
    interval. This might lead to overloading of the surrogate
    during a flash crowd.

7. Some implementations of bind can cause DNS timeouts to occur
    while handling exceptional situations. For example, timeouts
    can occur for NS redirections to unknown domains.

[3]. Transport-Layer Request-Routing

   At the transport-layer finer levels of granularity can be achieved
   by the close inspection of client's requests. In this approach, the
   Request-Routing system inspects the information available in the
   first packet of the client's request to make surrogate selection
   decisions. The inspection of the client's requests provides data
   about the client's IP address, port information, and layer 4
   protocol. The acquired data could be used in combination with
   user-defined policies and other metrics to determine the selection
   of a surrogate that is better suited to serve the request. The
   techniques that are used to hand off the session to a more
   appropriate surrogate are beyond the scope of this document.

   In general, the forward-flow traffic (client to newly selected
   surrogate) will flow through the surrogate originally chosen by DNS.
   The reverse-flow (surrogate to client) traffic, which normally
   transfers much more data than the forward flow, would typically
   take the direct path.

The overhead associated with transport-layer Request-Routing makes
it better suited for long-lived sessions such as FTP [1]and RTSP [3].
However, it also could be used to direct clients away from overloaded
surrogates.

In general, transport-layer Request-Routing can be combined with
DNS based techniques. As stated earlier, DNS based methods resolve
clients requests based on domains or sub domains with exposure to
the client's DNS server IP address. Hence, the DNS based methods
could be used as a first step in deciding on an appropriate
surrogate with more accurate refinement made by the
transport-layer Request-Routing system.

4. Application-Layer Request-Routing

Application-layer Request-Routing systems perform deeper
examination of client's packets beyond the transport layer
header. Deeper examination of client's packets provides
fine-grained Request-Routing control down to the level of
individual objects. The process could be performed in real
time at the time of the object request. The exposure to the
client's IP address combined with the fine-grained knowledge
of the requested objects enable application-layer Request-
Routing systems to provide better control over the selection of
the best surrogate.

4.1 Header Inspection

Applications such as HTTP [4], RTSP [3], and SSL [2] provide hints
in the initial portion of the session about how the client request
must be directed. These hints may come from the URL of the content
or other parts of the MIME request header such as Cookies.

4.1.1 URL-Based Request-Routing

A Uniform Resource Locator (URL) [7] consists of a naming scheme
followed by a string whose format is a function of the naming
scheme. The string must start with a constant prefix "URL:".
Within the URL of a object, the first element is the name of the
scheme, separated from the rest of the object by a colon. The rest
of the URL follows the colon in a format depending on the scheme.
For example, an FTP URL may optionally specify the FTP data transfer
type by which an object is to be retrieved. Most of the methods
correspond to the FTP Data Types such as ASCII and IMAGE for the
retrieval of a document, as specified in FTP by the "TYPE" Command.
The data type is specified by a suffix to the URL.

In a similar fashion, HTTP and RTSP content requests describe the
requested content by its URL. In many cases, this information

is sufficient to disambiguate the content and suitably direct the
request. In most cases, it may be sufficient to make Request-
Routing decision just by examining the prefix or suffix of the URL.

[4.1.1.1](#) 302 Redirection

In this approach, the client's request is first resolved to a virtual surrogate. Consequently, the surrogate returns an application-specific code such as the 302 (in the case of HTTP [[4](#)] or RTSP [[3](#)]) to redirect the client to the actual delivery node.

This technique is relatively simple to implement. However, the main drawback of this method is the additional latency involved in sending the redirect message back to the client.

[4.1.1.2](#) In-Path Element

In this technique, an In-Path element is present in the network in the forwarding path of the client's request. The In-Path element provides transparent interception of the transport connection. The In-Path element examines the client's content requests and performs Request-Routing decisions.

The In-Path element then splices the client connection to a connection with the appropriate delivery node and passes along the content request. In general, the return path would go through the In-Path element. However, it is possible to arrange for a direct return by passing the address translation information to the surrogate or delivery node through some proprietary means.

The primary disadvantage with this method is the performance implications of URL-parsing in the path of the network traffic. However, it is generally the case that the return traffic is much larger than the forward traffic.

The technique allows for the possibility of portioning the traffic among a set of delivery nodes by content objects identified by URLs. This allows object-specific control of server loading. For example, requests for non-cacheable object types may be directed away from a cache.

[4.1.2](#) Mime Header-Based Request-Routing

This technique involves the task of using MIME-headers such as Cookie, Language, and User-Agent, in order to select a surrogate.

Cookies can be used to identify a customer or session by a web site. Cookie based Request-Routing provides content service differentiation based on the client. This approach works provided that the cookies belong to the client. In addition, it is possible to direct a connection from a multi-session transaction to be directed to the same server to achieve session-level persistence.

   The language header can be used to direct traffic to a
   language-specific delivery node. The user-agent header helps
   identify the type of client device. For example, a voice-browser,
   PDA, or cell phone can indicate the type of delivery node that has
   content specialized to handle the content request.

4.1.3 Site-Specific Identifiers

   Site-specific identifiers help authenticate and identify a session
   from a specific user. This information may be used to direct a
   content request.

   An example of a site-specific identifier is the SSL Session
   Identifier. This identifier is generated by a web server and used
   by the web client in succeeding sessions to identify itself and
   avoid an entire security authentication exchange. In order to
   inspect the session identifier, an In-Path element would observe
   the responses of the web server and determine the session identifier
   which is then used to associate the session to a specific server.
   The remaining sessions are directed based on the stored
   session identifier.

4.2 Content Modification

   This technique enables a content provider to take direct control
   over Request-Routing decisions without the need for specific
   witching devices or directory services in the path between the
   client and the origin server. Basically, a content provider can
   directly communicate to the client the best surrogate that can
   serve the request. Decisions about the best surrogate can be made
   on a per-object basis or it can depend on a set of metrics. The
   overall goal is to improve scalability and the performance for
   delivering the modified content, including all embedded objects.

   In general, the method takes advantage of content objects that
   consist of basic structure that includes references to additional,
   embedded objects. For example, most web pages, consist of an HTML
   document that contains plain text together with some embedded objects,
   such as GIF or JPEG images. The embedded objects are referenced using
   embedded HTML directives. In general, embedded HTML directives direct
   the client to retrieve the embedded objects from the origin server.
   A content provider can now modify references to embedded objects
   such that they could be fetched from the best surrogate. This
   technique is also known as URL rewriting. The basic types of URL
   rewriting are discussed in the following subsections.

4.2.1 A-priori URL Rewriting

   In this scheme, a content provider rewrites the embedded URLs

before the content is positioned on the origin server. In this
case, URL rewriting can be done either manually or by using a
software tools that parse the content and replace embedded URLs.

A-priori URL rewriting alone does not allow consideration of client specifics for Request-Routing. However, it can be used in combination with DNS Request-Routing to direct related DNS queries into the domain name space of the service provider. Dynamic Request-Routing based on client specifics are then done using the DNS approach.

4.2.2 On-Demand URL Rewriting

On-Demand or dynamic URL rewriting, modifies the content when the client request reaches the origin server. At this time, the identity of the client is known and can be considered when rewriting the embedded URLs. In particular, an automated process can determine, on-demand, which surrogate would serve the requesting client best. The embedded URLs can then be rewritten to direct the client to retrieve the objects from the best surrogate rather than from the origin server.

4.2.3 Content Modification Limitations

Content modification as a Request-Routing mechanism suffers from the following limitations:

1. The first request from a client to a specific site must be served from the origin server.

2. Content that has been modified to include references to nearby surrogates rather than to the origin server should be marked as non-cacheable. Alternatively, such pages can be marked to be cacheable only for a relatively short period of time. Rewritten URLs on cached pages can cause problems, because they can get outdated and point to surrogates that are no longer available or no longer good choices.

5. Combination of Multiple Mechanisms

There are environments in which a combination of different mechanisms can be beneficial and advantageous over using one of the proposed mechanisms alone. The following example illustrates how the mechanisms can be used in combination.

A basic problem of DNS Request-Routing is the resolution granularity that allows resolution on a per-domain level only. A per-object redirection cannot easily be achieved. However, content modification can be used together with DNS Request-Routing to overcome this problem. With content modification, references to different objects on the same origin server can be rewritten to point into different domain name spaces. Using DNS Request-Routing, requests for those objects can now dynamically be directed to different surrogates.

6. Security Considerations

The main objective of this document is to provide a summary of current Request-Routing techniques. Such techniques are currently implemented in the Internet. The document acknowledges that security must be addressed by any entity that implements any technique that redirects client's requests. However, the details of such techniques are beyond the scope of this document.

7. Acknowledgements

   The authors acknowledge the contributions and comments of Ian Cooper (Equinix), Nalin Mistry (Nortel), Wayne Ding (Nortel) and Eric Dean (CrystalBall).

8. References

   [1] Postel, J., "File Transfer Protocol", RFC 765, June 1980,

   [2] Dierks, T. and C. Allen, "The TLS Protocol Version 1", RFC 2246, January 1999.

   [3] Schulzrinne, H., Rao, A. and R. Lanphier, "Real Time Streaming Protocol", RFC 2326, April 1998.

   [4] 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.

   [5] Day, M., Cain, B. and G. Tomlinson, " A Model for Content Internetworking (CDI)", http://www.ietf.org/internet-drafts/draft-day-cdnp-model-09.txt (Group Last Call).

   [6] Partridge, C., Mendez, T., Milliken W., "Host Anycasting Service", RFC 1546, November 1993.

   [7] Berners-Lee, T., Masinter, L., McCahill, M., "Uniform Resource Locator   (URL), RFC 1738,May 1994.

   [8] H. Schulzrinne, H., Fokus, G., Casner, S., Frederick, R., Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

9. Authors' Addresses

   Abbie Barbir
   Nortel Networks
   3500 Carling Avenue, Nepean Ontario K2H 8E9 Canada
   EMail: abbieb@nortelnetworks.com

Brad Cain
Cereva Networks
EMail: bcain@cereva.com

Fred Douglis
AT&T Labs
Room B137
180 Park Ave, Bldg 103
Florham Park, NJ 07932, US
EMail: douglis@research.att.com

Mark Green
CacheFlow
650 Almanor Avenue
Sunnyvale, CA 94085, US
EMail: markg@cacheflow.com

Markus Hofmann
Lucent Technologies
Room 4F-513
101 Crawfords Corner Rd.
Holmdel, NJ 07733, US
EMail: hofmann@bell-labs.com

Raj Nair
Cisco Systems
50 Nagog Park
Acton, MA 01720, US
EMail: rnair@cisco.com

Doug Potter
Cisco Systems
50 Nagog Park
Acton, MA 01720, US
EMail: dougpott@cisco.com

Oliver Spatscheck
AT&T Labs
Room B131
180 Park Ave, Bldg 103
Florham Park, NJ 07932, US
EMail: spatsch@research.att.com

Appendix A

Measurements

Request-Routing systems can use a variety of metrics in order
to determine the best surrogate that can serve a client's request.
In general, these metrics are based on network measurements and
feedback from surrogates. It is possible to combine multiple
metrics using both proximity and surrogate feedback for best
surrogate selection. The following sections describe several
well known metrics as well as the major techniques for obtaining
them.

A.1 Proximity Measurements

Proximity measurements can be used by the Request-Routing system to
direct users to the "closest" surrogate. In a DNS Request-Routing
system, the measurements are made to the client's local DNS server.
However, when the IP address of the client is accessible more
accurate proximity measurements can be obtained.

Furthermore, proximity measurements can be exchanged between
surrogates and the requesting entity. In many cases, proximity
measurements are "one-way" in that they measure either the forward
or reverse path of packets from the surrogate to the requesting
entity. This is important as many paths in the Internet are
asymmetric.

In order to obtain a set of proximity measurements, a network
may employ active probing techniques and/or passive measurement
techniques. The following sections describe these two techniques.

A.1.1 Active Probing

Active probing is when past or possible requesting entities are
probed using one or more techniques to determine one or more
metrics from each surrogate or set of surrogates. An example of
a probing technique is an ICMP ECHO Request that is periodically
sent from each surrogate or set of surrogates to a potential
requesting entity.

In any active probing approach, a list of potential requesting
entities need to be obtained. This list can be generated
dynamically. Here, as requests arrive, the requesting entity
addresses can be cached for later probing. Another potential
solution is to use an algorithm to divide address space into blocks
and to probe random addresses within those blocks. Limitations
of active probing techniques include:

    1. Measurements can only be taken periodically.

2. Firewalls and NATs disallow probes.

> 3. Probes often cause security alarms to be triggered on
>   intrusion detection systems.

A.1.2 Passive Measurement

Passive measurements could be obtained when a client performs data
transfers to or from a surrogate. Here, a bootstrap mechanism is
used to direct the client to a bootstrap surrogate. Once the client
connects, the actual performance of the transfer is measured. This
data is then fed back into the Request-Routing system.

An example of passive measurement is to watch the packet loss
from a client to a surrogate by observing TCP behavior. Latency
measurements can also be learned by observing TCP behavior. The
limitations of passive measurement approach are directly related
to the bootstrapping mechanism. Basically, a good mechanism is
needed to ensure that not every surrogate is tested per client
in order to obtain the data.

A.1.3 Metric Types

The following sections list some of the metrics, which can be used
for proximity calculations.

* Latency: Network latency measurements metrics are used to
  determine the surrogate (or set of surrogates) that has the
  least delay to the requesting entity. These measurements
  can be obtained using either an active probing approach or a
  passive network measurement system.

* Packet Loss: Packet loss measurements can be used as a
  selection metric. A passive measurement approach can easily
  obtain packet loss information from TCP header information.
  Active probing can periodically measure packet loss from
  probes.

* Hop Counts: Router hops from the surrogate to the requesting
  entity can be used as a proximity measurement.

* BGP Information: BGP AS PATH and MED attributes can be used
  to determine the "BGP distance" to a given prefix/length
  pair. In order to use BGP information for proximity
  measurements, it must be obtained at each surrogate
  site/location.

A.2 Surrogate Feedback

The Request-Routing system can use feedback from surrogates in
order to select a "least-loaded" delivery node. Feedback can
be delivered from each surrogate or can be aggregated by site or
by location.

A.2.1 Probing

Feedback information may be obtained by periodically probing a
surrogate by issuing an HTTP request and observing the behavior.
The problems with probing for surrogate information are:

    1. It is difficult to obtain "real-time" information.
    2. Non-real-time information may be inaccurate.

Consequently, feedback information can be obtained by agents that
reside on surrogates that can communicate a variety of metrics about
their nodes.

A.2.2 Well Known Metrics

The following provides a list of several of the popular metrics
that are used for surrogate feedback:

    *   Surrogate CPU Load.
    *   Interface Load / Dropped packets.
    *   Number of connections being served.
    *   Storage I/O Load.