Network Working Group
Request for Comments: 2187
Category: Informational

D. Wessels
K. Claffy
National Laboratory for Applied
Network Research/UCSD
September 1997

Application of Internet Cache Protocol (ICP), version 2

Status of this Memo

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

Abstract

This document describes the application of ICPv2 (Internet Cache Protocol version 2, RFC2186) to Web caching. ICPv2 is a lightweight message format used for communication among Web caches. Several independent caching implementations now use ICP[3,5], making it important to codify the existing practical uses of ICP for those trying to implement, deploy, and extend its use.

ICP queries and replies refer to the existence of URLs (or objects) in neighbor caches. Caches exchange ICP messages and use the gathered information to select the most appropriate location from which to retrieve an object. A companion document (RFC2186) describes the format and syntax of the protocol itself. In this document we focus on issues of ICP deployment, efficiency, security, and interaction with other aspects of Web traffic behavior.

Table of Contents

1. Introduction

ICP is a lightweight message format used for communicating among Web caches. ICP is used to exchange hints about the existence of URLs in neighbor caches. Caches exchange ICP queries and replies to gather information for use in selecting the most appropriate location from which to retrieve an object.

This document describes the implementation of ICP in software. For a description of the protocol and message format, please refer to the companion document (RFC2186). We avoid making judgments about whether or how ICP should be used in particular Web caching configurations. ICP may be a "net win" in some situations, and a "net loss" in others. We recognize that certain practices described in this document are suboptimal. Some of these exist for historical reasons. Some aspects have been improved in later versions. Since this document only serves to describe current practices, we focus on documenting rather than evaluating. However, we do address known security problems and other shortcomings.

The remainder of this document is written as follows. We first describe Web cache hierarchies, explain motivation for using ICP, and demonstrate how to configure its use in cache hierarchies. We then provide a step-by-step description of an ICP query-response transaction. We then discuss ICP interaction with firewalls, and briefly touch on multicasting ICP. We end with lessons with have learned during the protocol development and deployement thus far, and the canonical security considerations.

ICP was initially developed by Peter Danzig, et. al. at the University of Southern California as a central part of hierarchical caching in the Harvest research project[3].

2. Web Cache Hierarchies

A single Web cache will reduce the amount of traffic generated by the clients behind it. Similarly, a group of Web caches can benefit by sharing another cache in much the same way. Researchers on the Harvest project envisioned that it would be important to connect Web caches hierarchically. In a cache hierarchy (or mesh) one cache establishes peering relationships with its neighbor caches. There are two types of relationship: parent and sibling. A parent cache is essentially one level up in a cache hierarchy. A sibling cache is on the same level. The terms "neighbor" and "peer" are used to refer to either parents or siblings which are a single "cache-hop" away. Figure 1 shows a simple hierarchy configuration.

But what does it mean to be "on the same level" or "one level up?" The general flow of document requests is up the hierarchy. When a cache does not hold a requested object, it may ask via ICP whether any of its neighbor caches has the object. If any of the neighbors does have the requested object (i.e., a "neighbor hit"), then the cache will request it from them. If none of the neighbors has the object (a "neighbor miss"), then the cache must forward the request either to a parent, or directly to the origin server. The essential difference between a parent and sibling is that a "neighbor hit" may be fetched from either one, but a "neighbor miss" may NOT be fetched from a sibling. In other words, in a sibling relationship, a cache can only ask to retrieve objects that the sibling already has cached, whereas the same cache can ask a parent to retrieve any object regardless of whether or not it is cached. A parent cache's role is

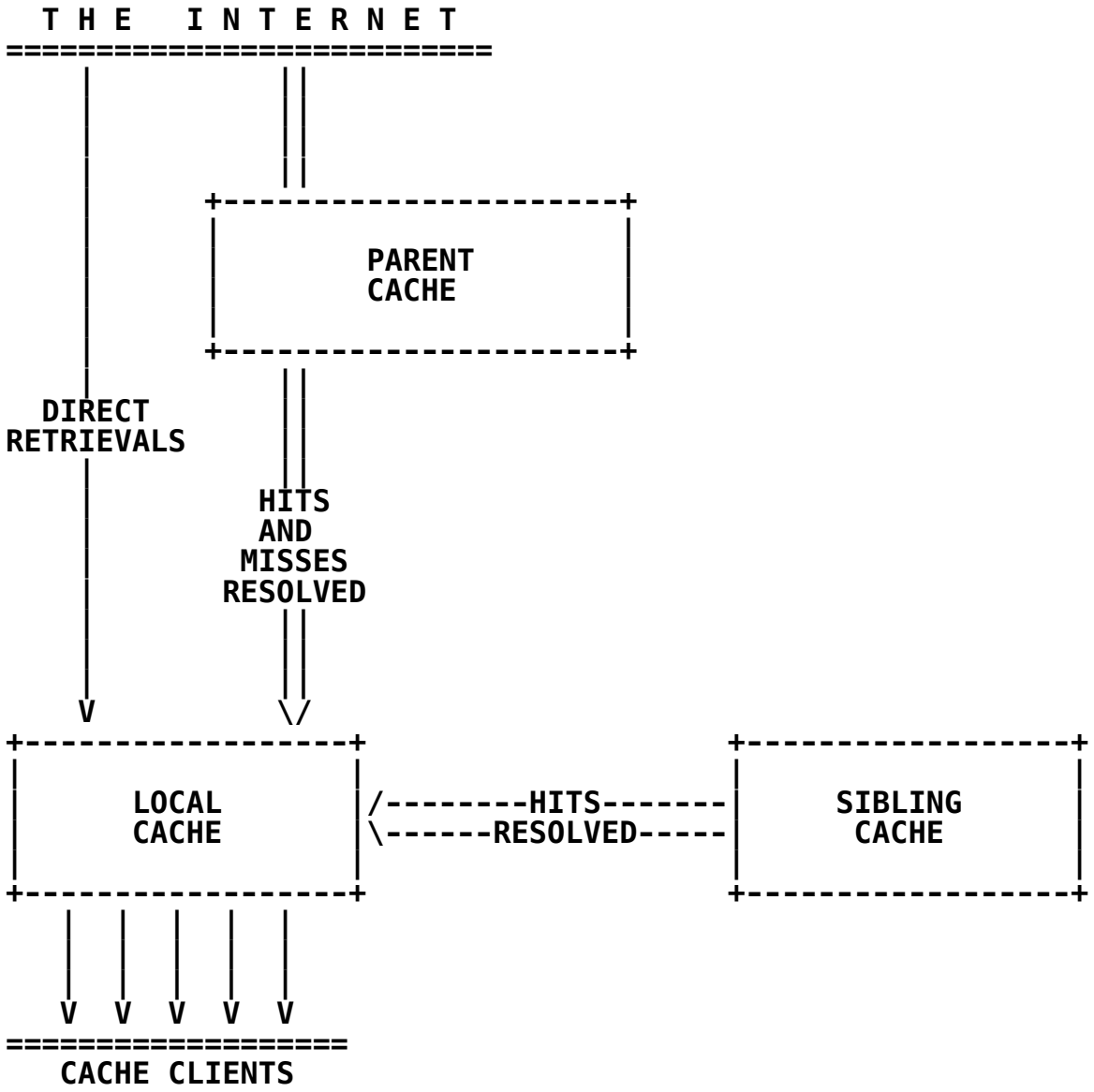

FIGURE 1: A Simple Web cache hierarchy. The local cache can retrieve hits from sibling caches, hits and misses from parent caches, and some requests directly from origin servers.

to provide "transit" for the request if necessary, and accordingly parent caches are ideally located within or on the way to a transit Internet service provider (ISP).

Squid and Harvest allow for complex hierarchical configurations. For example, one could specify that a given neighbor be used for only a certain class of requests, such as URLs from a specific DNS domain.

Additionally, it is possible to treat a neighbor as a sibling for some requests and as a parent for others.

The cache hierarchy model described here includes a number of features to prevent top-level caches from becoming choke points. One is the ability to restrict parents as just described previously (by domains). Another optimization is that the cache only forwards cachable requests to its neighbors. A large class of Web requests are inherently uncachable, including: requests requiring certain types of authentication, session-encrypted data, highly personalized responses, and certain types of database queries. Lower level caches should handle these requests directly rather than burdening parent caches.

3. What is the Added Value of ICP?

Although it is possible to maintain cache hierarchies without using ICP, the lack of ICP or something similar prohibits the existence of sibling meta-communicative relationships, i.e., mechanisms to query nearby caches about a given document.

One concern over the use of ICP is the additional delay that an ICP query/reply exchange contributes to an HTTP transaction. However, if the ICP query can locate the object in a nearby neighbor cache, then the ICP delay may be more than offset by the faster delivery of the data from the neighbor. In order to minimize ICP delays, the caches (as well as the protocol itself) are designed to return ICP requests quickly. Indeed, the application does minimal processing of the ICP request, most ICP-related delay is due to transmission on the network.

ICP also serves to provide an indication of neighbor reachability. If ICP replies from a neighbor fail to arrive, then either the network path is congested (or down), or the cache application is not running on the ICP-queried neighbor machine. In either case, the cache should not use this neighbor at this time. Additionally, because an idle cache can turn around the replies faster than a busy one, all other things being equal, ICP provides some form of load balancing.

4. Example Configuration of ICP Hierarchy

Configuring caches within a hierarchy requires establishing peering relationships, which currently involves manual configuration at both peering endpoints. One cache must indicate that the other is a parent or sibling. The other cache will most likely have to add the first cache to its access control lists.

Below we show some sample configuration lines for a hypothetical situation. We have two caches, one operated by an ISP, and another operated by a customer. First we describe how the customer would configure his cache to peer with the ISP. Second, we describe how the ISP would allow the customer access to its cache.

4.1. Configuring the `proxy.customer.org' cache

In Squid, to configure parents and siblings in a hierarchy, a `cache_host' directive is entered into the configuration file. The format is:

```
cache_host hostname type http-port icp-port [options]
```

Where type is either `parent', `sibling', or `multicast'. For our example, it would be:

```
cache_host cache.isp.com parent 8080 3130
```

This configuration will cause the customer cache to resolve most cache misses through the parent (`cgi-bin' and non-GET requests would be resolved directly). Utilizing the parent may be undesirable for certain servers, such as servers also in the customer.org domain. To always handle such local domains directly, the customer would add this to his configuration file:

```
local_domain customer.org
```

It may also be the case that the customer wants to use the ISP cache only for a specific subset of DNS domains. The need to limit requests this way is actually more common for higher levels of cache hierarchies, but it is illustrated here nonetheless. To limit the ISP cache to a subset of DNS domains, the customer would use:

```
cache_host_domain cache.isp.com com net org
```

Then, any requests which are NOT in the .com, .net, or .org domains would be handled directly.

4.2. Configuring the `cache.isp.com' cache

To configure the query-receiving side of the cache peer relationship one uses access lists, similar to those used in routing peers. The access lists support a large degree of customization in the peering relationship. If there are no access lines present, the cache allows the request by default.

Note that the cache.isp.com cache need not explicitly specify the customer cache as a peer, nor is the type of relationship encoded within the ICP query itself. The access control entries regulate the relationships between this cache and its neighbors. For our example, the ISP would use:

```
acl src Customer  proxy.customer.org
http_access allow Customer
icp_access  allow Customer
```

This defines an access control entry named `Customer' which specifies a source IP address of the customer cache machine. The customer cache would then be allowed to make any request to both the HTTP and ICP ports (including cache misses). This configuration implies that the ISP cache is a parent of the customer.

If the ISP wanted to enforce a sibling relationship, it would need to deny access to cache misses. This would be done as follows:

```
miss_access deny Customer
```

Of course the ISP should also communicate this to the customer, so that the customer will change his configuration from parent to sibling. Otherwise, if the customer requests an object not in the ISP cache, an error message is generated.

5. Applying the Protocol

The following sections describe the ICP implementation in the Harvest[3] (research version) and Squid Web cache[5] packages. In terms of version numbers, this means version 1.4pl2 for Harvest and version 1.1.10 for Squid.

The basic sequence of events in an ICP transaction is as follows:

1. Local cache receives an HTTP[1] request from a cache client.

2. The local cache sends ICP queries (section 5.1).

3. The peer cache(s) receive the queries and send ICP replies (section 5.2).

4. The local cache receives the ICP replies and decides where to forward the request (section 5.3).

5.1. Sending ICP Queries

5.1.1. Determine whether to use ICP at all

Not every HTTP request requires an ICP query to be sent. Obviously, cache hits will not need ICP because the request is satisfied immediately. For origin servers very close to the cache, we do not want to use any neighbor caches. In Squid and Harvest, the administrator specifies what constitutes a `local' server with the `local_domain' and `local_ip' configuration options. The cache always contacts a local server directly, never querying a peer cache.

There are other classes of requests that the cache (or the administrator) may prefer to forward directly to the origin server. In Squid and Harvest, one such class includes all non-GET request methods. A Squid cache can also be configured to not use peers for URLs matching the `hierarchy_stoplist'.

In order for an HTTP request to yield an ICP transaction, it must:

- not be a cache hit
- not be to a local server
- be a GET request, and
- not match the `hierarchy_stoplist' configuration.

We call this a "hierarchical" request. A "non-hierarchical" request is one that doesn't generate any ICP traffic. To avoid processing requests that are likely to lower cache efficiency, one can configure the cache to not consult the hierarchy for URLs that contain certain strings (e.g. `cgi_bin').

5.1.2. Determine which peers to query

By default, a cache sends an ICP_OP_QUERY message to each peer, unless any one of the following are true:

- Restrictions prevent querying a peer for this request, based on the configuration directive `cache_host_domain', which specifies a set of DNS domains (from the URLs) for which the peer should or should not be queried. In Squid, a more flexible directive ('cache_host_acl') supports restrictions on other parts of the request (method, port number, source, etc.).

- The peer is a sibling, and the HTTP request includes a "Pragma: no-cache" header. This is because the sibling would be asked to transit the request, which is not allowed.

- The peer is configured to never be sent ICP queries (i.e. with the `no-query' option).

If the determination yields only one queryable ICP peer, and the Squid configuration directive `single_parent_bypass' is set, then one can bypass waiting for the single ICP response and just send the HTTP request directly to the peer cache.

The Squid configuration option `source_ping' configures a Squid cache to send a ping to the original source simultaneous with its ICP queries, in case the origin is closer than any of the caches.

5.1.3. Calculate the expected number of ICP replies

Harvest and Squid want to maximize the chance to get a HIT reply from one of the peers. Therefore, the cache waits for all ICP replies to be received. Normally, we expect to receive an ICP reply for each query sent, except:

- When the peer is believed to be down. If the peer is down Squid and Harvest continue to send it ICP queries, but do not expect the peer to reply. When an ICP reply is again received from the peer, its status will be changed to up.

  The determination of up/down status has varied a little bit as the Harvest and Squid software evolved. Both Harvest and Squid mark a peer down when it fails to reply to 20 consecutive ICP queries. Squid also marks a peer down when a TCP connection fails, and up again when a diagnostic TCP connection succeeds.

- When sending to a multicast address. In this case we'll probably expect to receive more than one reply, and have no way to definitively determine how many to expect. We discuss multicast issues in section 7 below.

5.1.4. Install timeout event

Because ICP uses UDP as underlying transport, ICP queries and replies may sometimes be dropped by the network. The cache installs a timeout event in case not all of the expected replies arrive. By default Squid and Harvest use a two-second timeout. If object retrieval has not commenced when the timeout occurs, a source is selected as described in section 5.3.9 below.

5.2. Receiving ICP Queries and Sending Replies

When an ICP_OP_QUERY message is received, the cache examines it and decides which reply message is to be sent. It will send one of the following reply opcodes, tested for use in the order listed:

5.2.1. ICP_OP_ERR

The URL is extracted from the payload and parsed. If parsing fails, an ICP_OP_ERR message is returned.

5.2.2. ICP_OP_DENIED

The access controls are checked. If the peer is not allowed to make this request, ICP_OP_DENIED is returned. Squid counts the number of ICP_OP_DENIED messages sent to each peer. If more than 95% of more than 100 replies have been denied, then no reply is sent at all. This prevents misconfigured caches from endlessly sending unnecessary ICP messages back and forth.

5.2.3. ICP_OP_HIT

If the cache reaches this point without already matching one of the previous opcodes, it means the request is allowed and we must determine if it will be HIT or MISS, so we check if the URL exists in the local cache. If so, and if the cached entry is fresh for at least the next 30 seconds, we can return an ICP_OP_HIT message. The stale/fresh determination uses the local refresh (or TTL) rules.

Note that a race condition exists for ICP_OP_HIT replies to sibling peers. The ICP_OP_HIT means that a subsequent HTTP request for the named URL would result in a cache hit. We assume that the HTTP request will come very quickly after the ICP_OP_HIT. However, there is a slight chance that the object might be purged from this cache before the HTTP request is received. If this happens, and the replying peer has applied Squid's `miss_access' configuration then the user will receive a very confusing access denied message.

5.2.3.1. ICP_OP_HIT_OBJ

Before returning the ICP_OP_HIT message, we see if we can send an ICP_OP_HIT_OBJ message instead. We can use ICP_OP_HIT_OBJ if:

- The ICP_OP_QUERY message had the ICP_FLAG_HIT_OBJ flag set.

- The entire object (plus URL) will fit in an ICP message. The maximum ICP message size is 16 Kbytes, but an application may choose to set a smaller maximum value for ICP_OP_HIT_OBJ replies.

Normally ICP replies are sent immediately after the query is received, but the ICP_OP_HIT_OBJ message cannot be sent until the object data is available to copy into the reply message. For Squid and Harvest this means the object must be "swapped in" from disk if it is not already in memory. Therefore, on average, an ICP_OP_HIT_OBJ reply will have higher latency than ICP_OP_HIT.

5.2.4. ICP_OP_MISS_NOFETCH

At this point we have a cache miss. ICP has two types of miss replies. If the cache does not want the peer to request the object from it, it sends an ICP_OP_MISS_NOFETCH message.

5.2.5. ICP_OP_MISS

Finally, an ICP_OP_MISS reply is returned as the default. If the replying cache is a parent of the querying cache, the ICP_OP_MISS indicates an invitation to fetch the URL through the replying cache.

5.3. Receiving ICP Replies

Some ICP replies will be ignored; specifically, when any of the following are true:

- The reply message originated from an unknown peer.
- The object named by the URL does not exist.
- The object is already being fetched.

5.3.1. ICP_OP_DENIED

If more than 95% of more than 100 replies from a peer cache have been ICP_OP_DENIED, then such a high denial rate most likely indicates a configuration error, either locally or at the peer. For this reason, no further queries will be sent to the peer for the duration of the cache process.

5.3.2. ICP_OP_HIT

Object retrieval commences immediately from the replying peer.

5.3.3. ICP_OP_HIT_OBJ

The object data is extracted from the ICP message and the retrieval is complete. If there is some problem with the ICP_OP_HIT_OBJ message (e.g. missing data) the reply will be treated like a standard ICP_OP_HIT.

5.3.4. ICP_OP_SECHO

Object retrieval commences immediately from the origin server because the ICP_OP_SECHO reply arrived prior to any ICP_OP_HIT's. If an ICP_OP_HIT had arrived prior, this ICP_OP_SECHO reply would be ignored because the retrieval has already started.

5.3.5. ICP_OP_DECHO

An ICP_OP_DECHO reply is handled like an ICP_OP_MISS. Non-ICP peers must always be configured as parents; a non-ICP sibling makes no sense. One serious problem with the ICP_OP_DECHO feature is that since it bounces messages off the peer's UDP echo port, it does not indicate that the peer cache is actually running -- only that network connectivity exists between the pair.

5.3.6. ICP_OP_MISS

If the peer is a sibling, the ICP_OP_MISS reply is ignored. Otherwise, the peer may be "remembered" for future use in case no HIT replies are received later (section 5.3.9).

Harvest and Squid remember the first parent to return an ICP_OP_MISS message. With Squid, the parents may be weighted so that the "first parent to miss" may not actually be the first reply received. We call this the FIRST_PARENT_MISS. Remember that sibling misses are entirely ignored, we only care about misses from parents. The parent miss RTT's can be weighted because sometimes the closest parent is not the one people want to use.

Also, recent versions of Squid may remember the parent with the lowest RTT to the origin server, using the ICP_FLAG_SRC_RTT option. We call this the CLOSEST_PARENT_MISS.

5.3.7. ICP_OP_MISS_NOFETCH

This reply is essentially ignored. A cache must not forward a request to a peer that returns ICP_OP_MISS_NOFETCH.

5.3.8. ICP_OP_ERR

Silently ignored.

5.3.9. When all peers MISS.

For ICP_OP_HIT and ICP_OP_SECHO the request is forwarded immediately. For ICP_OP_HIT_OBJ there is no need to forward the request. For all other reply opcodes, we wait until the expected number of replies have been received. When we have all of the expected replies, or when the query timeout occurs, it is time to forward the request.

Since MISS replies were received from all peers, we must either select a parent cache or the origin server.

- If the peers are using the ICP_FLAG_SRC_RTT feature, we forward the request to the peer with the lowest RTT to the origin server. If the local cache is also measuring RTT's to origin servers, and is closer than any of the parents, the request is forwarded directly to the origin server.

- If there is a FIRST_PARENT_MISS parent available, the request will be forwarded there.

- If the ICP query/reply exchange did not produce any appropriate parents, the request will be sent directly to the origin server (unless firewall restrictions prevent it).

5.4. ICP Options

The following options were added to Squid to support some new features while maintaining backward compatibility with the Harvest implementation.

5.4.1. ICP_FLAG_HIT_OBJ

This flag is off by default and will be set in an ICP_OP_QUERY message only if these three criteria are met:

- It is enabled in the cache configuration file with `udp_hit_obj on'.

- The peer must be using ICP version 2.

- The HTTP request must not include the "Pragma: no-cache" header.

5.4.2. ICP_FLAG_SRC_RTT

This flag is off by default and will be set in an ICP_OP_QUERY message only if these two criteria are met:

- It is enabled in the cache configuration file with `query_icmp on'.
- The peer must be using ICP version 2.

6. Firewalls

Operating a Web cache behind a firewall or in a private network poses some interesting problems. The hard part is figuring out whether the cache is able to connect to the origin server. Harvest and Squid provide an `inside_firewall' configuration directive to list DNS domains on the near side of a firewall. Everything else is assumed to be on the far side of a firewall. Squid also has a `firewall_ip' directive so that inside hosts can be specified by IP addresses as well.

In a simple configuration, a Squid cache behind a firewall will have only one parent cache (which is on the firewall itself). In this case, Squid must use that parent for all servers beyond the firewall, so there is no need to utilize ICP.

In a more complex configuration, there may be a number of peer caches also behind the firewall. Here, ICP may be used to check for cache hits in the peers. Occasionally, when ICP is being used, there may not be any replies received. If the cache were not behind a firewall, the request would be forwarded directly to the origin server. But in this situation, the cache must pick a parent cache, either randomly or due to configuration information. For example, Squid allows a parent cache to be designated as a default choice when no others are available.

7. Multicast

For efficient distribution, a cache may deliver ICP queries to a multicast address, and neighbor caches may join the multicast group to receive such queries.

Current practice is that caches send ICP replies only to unicast addresses, for several reasons:

- Multicasting ICP replies would not reduce the number of packets sent.

- It prevents other group members from receiving unexpected replies.
- The reply should follow unicast routing paths to indicate (unicast) connectivity between the receiver and the sender since the subsequent HTTP request will be unicast routed.

Trust is an important aspect of inter-cache relationships. A Web cache should not automatically trust any cache which replies to a multicast ICP query. Caches should ignore ICP messages from addresses not specifically configured as neighbors. Otherwise, one could easily pollute a cache mesh by running an illegitimate cache and having it join a group, return ICP_OP_HIT for all requests, and then deliver bogus content.

When sending to multicast groups, cache administrators must be careful to use the minimum multicast TTL required to reach all group members. Joining a multicast group requires no special privileges and there is no way to prevent anyone from joining "your" group. Two groups of caches utilizing the same multicast address could overlap, which would cause a cache to receive ICP replies from unknown neighbors. The unknown neighbors would not be used to retrieve the object data, but the cache would constantly receive ICP replies that it must always ignore.

To prevent an overlapping cache mesh, caches should thus limit the scope of their ICP queries with appropriate TTLs; an application such as mtrace[6] can determine appropriate multicast TTLs.

As mentioned in section 5.1.3, we need to estimate the number of expected replies for an ICP_OP_QUERY message. For unicast we expect one reply for each query if the peer is up. However, for multicast we generally expect more than one reply, but have no way of knowing exactly how many replies to expect. Squid regularly (every 15 minutes) sends out test ICP_OP_QUERY messages to only the multicast group peers. As with a real ICP query, a timeout event is installed and the replies are counted until the timeout occurs. We have found that the received count varies considerably. Therefore, the number of replies to expect is calculated as a moving average, rounded down to the nearest integer.

8. Lessons Learned

8.1. Differences Between ICP and HTTP

ICP is notably different from HTTP. HTTP supports a rich and sophisticated set of features. In contrast, ICP was designed to be simple, small, and efficient. HTTP request and reply headers consist of lines of ASCII text delimited by a CRLF pair, whereas ICP uses a fixed size header and represents numbers in binary. The only thing ICP and HTTP have in common is the URL.

Note that the ICP message does not even include the HTTP request method. The original implementation assumed that only GET requests would be cachable and there would be no need to locate non-GET requests in neighbor caches. Thus, the current version of ICP does not accommodate non-GET requests, although the next version of this protocol will likely include a field for the request method.

HTTP defines features that are important for caching but not expressible with the current ICP protocol. Among these are Pragma: no-cache, If-Modified-Since, and all of the Cache-Control features of HTTP/1.1. An ICP_OP_HIT_OBJ message may deliver an object which may not obey all of the request header constraints. These differences between ICP and HTTP are the reason we discourage the use of the ICP_OP_HIT_OBJ feature.

8.2. Parents, Siblings, Hits and Misses

Note that the ICP message does not have a field to indicate the intent of the querying cache. That is, nowhere in the ICP request or reply does it say that the two caches have a sibling or parent relationship. A sibling cache can only respond with HIT or MISS, not "you can retrieve this from me" or "you can not retrieve this from me." The querying cache must apply the HIT or MISS reply to its local configuration to prevent it from resolving misses through a sibling cache. This constraint is awkward, because this aspect of the relationship can be configured only in the cache originating the requests, and indirectly via the access controls configured in the queried cache as described earlier in section 4.2.

8.3. Different Roles of ICP

There are two different understandings of what exactly the role of ICP is in a cache mesh. One understanding is that ICP's role is only object location, specifically, to provide hints about whether or not a named object exists in a neighbor cache. An implied assumption is that cache hits are highly desirable, and ICP is used to maximize the chance of getting them. If an ICP message is lost due to congestion, then nothing significant is lost; the request will be satisfied regardless.

ICP is increasingly being tasked to fill a more complex role: conveying cache usage policy. For example, many organizations (e.g. universities) will install a Web cache on the border of their network. Such organizations may be happy to establish sibling relationships with other, nearby caches, subject to the following terms:

- Any of the organization's customers or users may request any object (cached or not).
- Anyone may request an object already in the cache.
- Anyone may request any object from the organization's servers behind the cache.
- All other requests are denied; specifically, the organization will not provide transit for requests in which neither the client nor the server falls within its domain.

To successfully convey policy the ICP exchange must very accurately predict the result (hit, miss) of a subsequent HTTP request. The result may often depend on other request fields, such as Cache-Control. So it's not possible for ICP to accurately predict the result without more, or perhaps all, of the HTTP request.

8.4. Protocol Design Flaws of ICPv2

We recognize certain flaws with the original design of ICP, and make note of them so that future versions can avoid the same mistakes.

- The NULL-terminated URL in the payload field requires stepping through the message an octet at a time to find some of the fields (i.e. the beginning of object data in an ICP_OP_HIT_OBJ message).

- Two fields (Sender Host Address and Requester Host Address) are IPv4 specific. However, neither of these fields are used in practice; they are normally zero-filled. If IP addresses have a role in the ICP message, there needs to be an address family descriptor for each address, and clients need to be able to say whether they want to hear IPv6 responses or not.

- Options are limited to 32 option flags and 32 bits of option data. This should be more like TCP, with an option descriptor followed by option data.

- Although currently used as the cache key, the URL string no longer serves this role adequately. Some HTTP responses now vary according to the requestor's User-Agent and other headers. A cache key must incorporate all non-transport headers present in the client's request. All non-hop-by-hop request headers should be sent in an ICP query.

- ICPv2 uses different opcode values for queries and responses. ICP should use the same opcode for both sides of a two-sided transaction, with a "query/response" indicator telling which side is which.

- ICPv2 does not include any authentication fields.

9. Security Considerations

Security is an issue with ICP over UDP because of its connectionless nature. Below we consider various vulnerabilities and methods of attack, and their implications.

Our first line of defense is to check the source IP address of the ICP message, e.g. as given by recvfrom(2). ICP query messages should be processed if the access control rules allow the querying address access to the cache. However, ICP reply messages must only be accepted from known neighbors; a cache must ignore replies from unknown addresses.

Because we trust the validity of an address in an IP packet, ICP is susceptible to IP address spoofing. In this document we address some consequences of IP address spoofing. Normally, spoofed addresses can only be detected by routers, not by hosts. However, the IP Authentication Header[7,8] can be used underneath ICP to provide cryptographic authentication of the entire IP packet containing the ICP protocol, thus eliminating the risk of IP address spoofing.

9.1. Inserting Bogus ICP Queries

Processing an ICP_OP_QUERY message has no known security implications, so long as the requesting address is granted access to the cache.

9.2. Inserting Bogus ICP Replies

Here we are concerned with a third party generating ICP reply messages which are returned to the querying cache before the real reply arrives, or before any replies arrive. The third party may insert bogus ICP replies which appear to come from legitimate neighbors. There are three vulnerabilities:

- Preventing a certain neighbor from being used

  If a third-party could send an ICP_OP_MISS_NOFETCH reply back before the real reply arrived, the (falsified) neighbor would not be used.

  A third-party could blast a cache with ICP_OP_DENIED messages until the threshold described in section 5.3.1 is reached, thereby causing the neighbor relationship to be temporarily terminated.

- Forcing a certain neighbor to be used

  If a third-party could send an ICP_OP_HIT reply back before the real reply arrived, the (falsified) neighbor would be used. This may violate the terms of a sibling relationship; ICP_OP_HIT replies mean a subsequent HTTP request will also be a hit.

  Similarly, if bogus ICP_OP_SECHO messages can be generated, the cache would retrieve requests directly from the origin server.

- Cache poisoning

  The ICP_OP_HIT_OBJ message is especially sensitive to security issues since it contains actual object data. In combination with IP address spoofing, this option opens up the likely possibility of having the cache polluted with invalid objects.

9.3. Eavesdropping

Multicasting ICP queries provides a very simple method for others to "snoop" on ICP messages. If enabling multicast, cache administrators should configure the application to use the minimum required multicast TTL, using a tool such as mtrace[6]. Note that the IP Encapsulating Security Payload [7,9] mechanism can be used to provide protection against eavesdropping of ICP messages.

Eavesdropping on ICP traffic can provide third parties with a list of URLs being browsed by cache users. Because the Requestor Host Address is zero-filled by Squid and Harvest, the URLs cannot be mapped back to individual host systems.

By default, Squid and Harvest do not send ICP messages for URLs containing `cgi-bin' or `?'. These URLs sometimes contain sensitive information as argument parameters. Cache administrators need to be aware that altering the configuration to make ICP queries for such URLs may expose sensitive information to outsiders, especially when multicast is used.

9.4. Blocking ICP Messages

Intentionally blocked (or discarded) ICP queries or replies will appear to reflect link failure or congestion, and will prevent the use of a neighbor as well as lead to timeouts (see section 5.1.4). If all messages are blocked, the cache will assume the neighbor is down and remove it from the selection algorithm. However, if, for example, every other query is blocked, the neighbor will remain "alive," but every other request will suffer the ICP timeout.

9.5. Delaying ICP Messages

The neighbor selection algorithm normally waits for all ICP MISS replies to arrive. Delaying queries or replies, so that they arrive later than they normally would, will cause additional delay for the subsequent HTTP request. Of course, if messages are delayed so that they arrive after the timeout, the behavior is the same as "blocking" above.

9.6. Denial of Service

A denial-of-service attack, where the ICP port is flooded with a continuous stream of bogus messages has three vulnerabilities:

- The application may log every bogus ICP message and eventually fill up a disk partition.

- The socket receive queue may fill up, causing legitimate messages to be dropped.
- The host may waste some CPU cycles receiving the bogus messages.

9.7. Altering ICP Fields

Here we assume a third party is able to change one or more of the ICP reply message fields.

Opcode

Changing the opcode field is much like inserting bogus messages described above. Changing a hit to a miss would prevent the peer from being used. Changing a miss to a hit would force the peer to be used.

Version

Altering the ICP version field may have unpredictable consequences if the new version number is recognized and supported. The receiving application should ignore messages with invalid version numbers. At the time of this writing, both version numbers 2 and 3 are in use. These two versions use some fields (e.g. Options) in a slightly different manner.

Message Length

An incorrect message length should be detected by the receiving application as an invalid ICP message.

Request Number

The request number is often used as a part of the cache key. Harvest does not use the request number. Squid uses the request number in conjunction with the URL to create a cache key. Altering the request number will cause a lookup of the cache key to fail. This is similar to blocking the ICP reply altogether.

There is no requirement that a cache use both the URL and the request number to locate HTTP requests with outstanding ICP queries (however both Squid and Harvest do). The request number must always be the same in the query and the reply. However, if the querying cache uses only the request number to locate pending requests, there is some possibility that a replying cache might increment the request number in the reply to give the false impression that the two caches are closer than they really are. In other words, assuming that there are a few ICP requests "in flight" at any given time, incrementing the reply request number trick the querying cache into seeing a smaller round-trip time than really exists.

Options

There is little risk in having the Options bitfields altered. Any option bit must only be set in a reply if it was also set in a query. Changing a bit from clear to set is detectable by the querying cache, and such a message must be ignored. Changing a bit from set to clear is allowed and has no negative side effects.

Option Data

ICP_FLAG_SRC_RTT is the only option which uses the Option Data field. Altering the RTT values returned here can affect the neighbor selection algorithm, either forcing or preventing the use of a neighbor.

URL

The URL and Request Number are used to generate the cache key. Altering the URL will cause a lookup of the cache key to fail, and the ICP reply to be entirely ignored. This is similar to blocking the ICP reply altogether.

9.8. Summary

- ICP_OP_HIT_OBJ is particularly vulnerable to security problems because it includes object data. For this, and other reasons, its use is discouraged.

- Falsifying, altering, inserting, or blocking ICP messages can cause an HTTP request to fail only in two situations:

  - If the cache is behind a firewall and cannot directly connect to the origin server.

- If a false ICP_OP_HIT reply causes the HTTP request to be forwarded to a sibling, where the request is a cache miss and the sibling refuses to continue forwarding the request on behalf of the originating cache.

o Falsifying, altering, inserting, or blocking ICP messages can easily cause HTTP requests to be forwarded (or not forwarded) to certain neighbors. If the neighbor cache has also been compromised, then it could serve bogus content and pollute a cache hierarchy.

o Blocking or delaying ICP messages can cause HTTP request to be further delayed, but still satisfied.

10. References

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

[2] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, CERN, Xerox PARC, University of Minnesota, December 1994.

[3] Bowman M., Danzig P., Hardy D., Manber U., Schwartz M., and Wessels D., "The Harvest Information Discovery and Access System", Internet Research Task Force - Resource Discovery, http://harvest.transarc.com/.

[4] Wessels D., Claffy K., "ICP and the Squid Web Cache", National Laboratory for Applied Network Research, http://www.nlanr.net/~wessels/Papers/icp-squid.ps.gz.

[5] Wessels D., "The Squid Internet Object Cache", National Laboratory for Applied Network Research, http://squid.nlanr.net/Squid/

[6] mtrace, Xerox PARC, ftp://ftp.parc.xerox.com/pub/net-research/ipmulti/.

[7] Atkinson, R., "Security Architecture for the Internet Protocol", RFC 1825, NRL, August 1995.

[8] Atkinson, R., "IP Authentication Header", RFC 1826, NRL, August 1995.

[9] Atkinson, R., "IP Encapsulating Security Payload (ESP)", RFC 1827, NRL, August 1995.

11. Acknowledgments

The authors wish to thank Paul A Vixie <paul@vix.com> for providing excellent feedback on this document, Martin Hamilton <martin@mrrl.lut.ac.uk> for pushing the development of multicast ICP, Eric Rescorla <ekr@terisa.com> and Randall Atkinson <rja@home.net> for assisting with security issues, and especially Allyn Romanow for keeping us on the right track.

12. Authors' Addresses

Duane Wessels
National Laboratory for Applied Network Research
10100 Hopkins Drive
La Jolla, CA 92093

EMail: wessels@nlanr.net

K. Claffy
National Laboratory for Applied Network Research
10100 Hopkins Drive
La Jolla, CA 92093

EMail: kc@nlanr.net