Independent Submission
Request for Comments: 6804
Category: Historic
ISSN: 2070-1721

B. Manning

November 2012

DISCOVER: Supporting Multicast DNS Queries

Abstract

This document describes the DISCOVER opcode, an experimental extension to the Domain Name System (DNS) to use multicast queries for resource discovery. This opcode was tested in experiments run during 1995 and 1996 for the Topology Based Domain Search (TBDS) project. This project is no longer active and there are no current plans to restart it. TBDS was the first known use of multicast transport for DNS. A client multicasts a DNS query using the DISCOVER opcode and processes the multiple responses that may result.

Status of This Memo

Copyright Notice

1. Introduction

The TBDS project developed extensions to existing network services to enable software for clients and servers of an application to become more resilient to changes in topology by dynamically sensing changes and switching between client/server and peer-peer methods for both end-system-to-server and server-to-server communications.

The first existing network service to be investigated was the Domain Name Systems (DNS), which is used to map symbolic Internet names to numeric Internet addresses. Based upon a hierarchical tree structure, the DNS relies upon uninterrupted connectivity of nodes to a special set of static, manually configured root servers. To improve the robustness and availability of the DNS service, TBDS developed and defined enhancements that enable nodes to map names to numbers without the need for uninterrupted connectivity to the Internet root servers. These techniques were automated, allowing transition between connected and unconnected operations to be done without direct human intervention.

These enhancements to the DNS server code are based on the open source BIND to support reception and processing of multicast packets.

Proof-of-concept modifications to BIND 8.1.2 were made to show that multicast awareness could be added to BIND. An analysis was made of the existing DNS code deployment and the schedule of new feature deployment so that we could synchronize TBDS with a more appropriate code base. Testing identified a race condition due to overloading the semantics of the DNS opcode that was used to communicate to servers.

This race condition was explored within the IETF regarding use of existing DNS opcodes. Discussion within the team and with others in the IETF led to the idea that we needed a new opcode that would not overload the semantics of existing opcodes. The original DNS design specification presumes that few clients exist that would share common DNS data. To correct this problem, a new opcode was designed to disambiguate TBDS requests from normal nameserver requests.

In the standard Domain Name System (DNS) [1] [2], queries are always unicast using the QUERY opcode. The TBDS research project [5], funded under DARPA grant F30602-99-1-0523, explored the use of multicast DNS [1] [2] queries for resource discovery by autonomous, mobile nodes in disconnected networks. The operations model is covered in the TBDS documentation. Multicast queries may return multiple replies, while the standard DNS QUERY operation (see Sections 3.7, 4.3, and 5 of RFC 1034 [1]; and Section 4.1.1 of RFC 1035 [2]) expects a single reply. Instead of extending the QUERY opcode, the project developed and tested a new query operation, DISCOVER, that was designed to accommodate multiple responses from a multicast query. The ability to accept multiple replies provides a basis for discrimination of man-in-the-middle attacks, which succeed by being the first to respond. Use of DISCOVER requires the use of caching in the receiver, so the ephemeral nature of stub resolvers is precluded.

This memo documents the processing rules for DISCOVER, for possible incorporation in a future revision of the DNS specification.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [3].

2. DISCOVER Processing Rules

A requester will send a DISCOVER query message to a multicast destination address, with some particular multicast scope. The requester must be prepared to receive multiple replies from multiple responders, although we expect that there will be a single reply per responder.

DISCOVER responses (i.e., response messages from DISCOVER queries) have standard Answer, Authority, and Additional sections. For example, the DISCOVER response is the same as the response to a QUERY operation. Zero-content answers should not be sent, to avoid badly formed or unfulfilled requests. Responses should be sent to the unicast address of the requester, and the source address should reflect the unicast address of the responder. DISCOVER responses may echo the request's Question section or leave it blank, just as for QUERY.

DISCOVER works like QUERY, with the following exceptions:

1. The Question section of a DISCOVER operation contains <QNAME=zonename,QTYPE=SOA> tuples, if the section is present.

   Within TBDS, this structure was augmented with: <QNAME=service,QTYPE=SRV>. While this worked, it would be cleaner to ask the SRV question in a separate pass; any future work should take this into consideration.

2. If QDCOUNT equals 0, then only servers willing to do recursion should answer; other servers must silently discard a DISCOVER request with QDCOUNT equals 0.

3. If QDCOUNT is not equal to 0, then only servers that are authoritative for the zones named by some QNAME should answer.

Hence, replies to DISCOVER queries will always be authoritative or else have RA (Recursion Available) set.

3. Using DISCOVER Queries

3.1. Performing Host Lookups

To perform a hostname lookup using DISCOVER, a node could:

- o Compute the zone name of the enclosing in-addr.arpa, ip6.int, or ip6.arpa domain.

- o DISCOVER whether any in-scope server(s) are authoritative for this zone.

  If so, query these authoritative servers for local in-addr/ip6 names.

- o If not, DISCOVER whether there are recursive servers available.

  If so, query these recursive servers for local in-addr/ip6 names.

  The requester can determine from the replies whether there are any DNS servers that are authoritative (or support recursion) for the zone.

- o Once the host's Fully Qualified Domain Name (FQDN) is known, repeat the process to discover the closest enclosing authoritative server for this local name.

- o Cache all NS and A data learned in this process, respecting Times To Live (TTLs).

3.2. Performing Service Lookups

To lookup a service name using DISCOVER, the following steps may be used:

- o Use DISCOVER as outlined in Section 3.1 to perform gethostbyaddr() and then gethostbyname() on one's own link-local address. This gives a list of local authoritative servers.

- Assume that the closest enclosing zone for which an authoritative server responds to an in-scope DISCOVER message is this host's "parent domain", and compute the SRV name as

  ```
  _service._transport.*.parentdomain.
  ```

  This is a change to the definition provided in RFC 1034 [1]. A wildcard label ("*") in the QNAME used in a DNS message with the opcode DISCOVER should be evaluated with special rules: the wildcard should match any label for which the DNS server data is authoritative. For example 'x.*.example.com.' would match 'x.y.example.com.' and 'x.yy.example.com.', provided that the server was authoritative for 'example.com.'

- Finally, send an SRV query for this SRV name to the discovered local authoritative servers to complete the getservbyname() call.

  This call returns a structure that can be populated by response values, as follows:

s_name	The name of the service, "_service" without the preceding underscore.
s_aliases	The names returned in the SRV Resource Records (RRs) in replies to the query.
s_port	The port number in the SRV RRs replies to the query. If these port numbers do not match, one of the port numbers is chosen, and only those names that correspond are returned.
s_proto	The transport protocol passed from the DNS process using the "_transport" label, without the preceding underscore.

3.3. Using DISCOVER for Disconnected Names

DISCOVER allows discovery of a host (for example, a printer offering LPD services) whose DNS server answers authoritatively for a domain name that hasn't been delegated to it, but is defined within some local scope. Since DISCOVER is explicitly defined to discover undelegated zones for tightly scoped queries, this behavior isn't a violation of DNS's coherency principles. Note that a responder to DISCOVER might not be traditional DNS software, it could be special-purpose software.

DISCOVER usage for disconnected networks with no authoritative servers can be achieved using the following conditions:

- Hosts run a "stub authoritative server" that acts as though its FQDN were a zone name.

- The computed SOA gives the host's FQDN as the MNAME, "." as the ANAME, seconds-since-1Jan2000 as the SERIAL, and low constants for EXPIRE and the other SOA timers.

- NS is used as the host's FQDN.

- The glue is computed as the host's link-local address, or hosts may run a "DNS stub server" that acts as though its FQDN were a zone name.

The rules governing the behavior of this server consist of a single change to the method of use, and no change whatsoever to the current format of DNS packets. Specifically, this extension allows UDP DNS queries, as documented in RFC 1035, Sections 4.1.1, 4.1.2, and 4.2.1, to be addressed to port 53 of statically assigned relative offset -4 within the range of multicast addresses defined as "administratively scoped" by Section 9 of RFC 2365 [6]. Within the full /8 of administratively scoped addresses, this corresponds to the destination address 239.255.255.251. Until the Multicast-Scope Zone Announcement Protocol (MZAP) or a similar protocol is implemented to allow hosts to discover the extent of the local multicast scopes that enclose them, it is anticipated that implementations will simply utilize the destination address 239.255.255.251. Queries sent via multicast MUST NOT request recursion.

In order to receive multicasted queries, DNS server implementations MUST listen on the -4 offset to their local scope (as above, in the absence of a method of determining the scope, this will be assumed to be relative to the full /8 allocated for administratively scoped multicast use, or 239.255.255.251) and respond via ordinary unicast UDP to ONLY those queries for which they have a positive answer that originated within a locally-configured zone file. That is, a server MUST NOT answer a multicasted query with cached information that it received from another server, nor may it request further resolution from other servers on behalf of a multicasted query. A multicast-capable server may, however, utilize multicast queries to perform further resolution on behalf of queries received via ordinary unicast. This is referred to as "proxy" operation. Multicast-enabled DNS servers MUST answer multicasted queries non-authoritatively. That is, when responding to a query that was received via multicast, they MUST NOT include an NS record that contains data that resolves back to their own IP address and MUST NOT set the AA bit.

Resolvers MUST anticipate receiving no replies to some multicasted queries, in the event that no multicast-enabled DNS server implementations are active within the local scope, or in the event that no positive responses exist to the transmitted query. That is, a query for the MX record for host.domain.com would go unanswered if no local server was able to resolve that request, if no MX record exists for host.domain.com, or if no local servers were capable of receiving multicast queries. The resolver that initiated the query MUST treat such non-response as a non-cacheable negative response. Since this multicast transmission does not provide reliable delivery, resolvers MAY repeat the transmission of a query in order to assure themselves that is has been received by any hosts capable of answering; however, any resolvers that repeat a query MUST increase the interval by a factor of two between each repetition. It is more likely, however, that any repeated queries will be performed under the explicit direction of the application driving the query, rather than autonomously by the resolver implementation.

It will often be the case that multicast queries will result in responses from multiple servers. In the event that the multicast query was generated via a current API such as gethostbyname, or as the result of a proxy operation, the first response received must be passed to the requesting application or host, and all subsequently received responses must be discarded. Future multicast-aware APIs that use DISCOVER should anticipate receiving multiple independent RR sets in response to queries and using external heuristics for selecting the most appropriate RR set.

Such servers should answer DISCOVER packets for its zone, and will be found by the iterative "discover closest enclosing authority server" by DISCOVER clients, in either the gethostbyname() or SRV cases described above. Note that stub servers answer only with zone names that exactly match QNAME's, not with zone names that are owned by QNAME's.

4. IANA Considerations

At such time as this idea might be considered for a future addition to the DNS protocol, IANA would need to assign a value for the opcode.

5. Security Considerations

The following paragraph on security considerations was written very early in the use and exploration of IP multicast and, as such, represents a fairly naive view on the type and scope of exploits that are enabled through the use of IP multicast. A more up-to-date understanding of multicast security considerations may be found in RFC 5294 [4].

No new security considerations are known to be introduced with a new DNS query operation. However, using multicast for service discovery has the potential for denial of service from flooding attacks. How to scope multicast is not part of the DISCOVER processing rules. It may also be possible to enable deliberate misconfiguration of clients simply by running a malicious DNS server that falsely claims to be authoritative for delegations. One possible way to mitigate this threat is to use credentials, such as CERT resource records within an RR set. The TBDS project took this approach. TBDS did not directly utilize DNS Security (DNSSEC), so possible interactions with DNSSEC-aware/-capable servers are unknown.

6. Acknowledgments

This material was generated in discussions on the mdns mailing list hosted by Zocalo in March 2000 and updated by discussions in September/October 2003 on a closed mailing list. David Lawrence, Scott Rose, Stuart Cheshire, Bill Woodcock, and Erik Guttman were active contributors. Suzanne Woolf was part of the original implementation team and an invaluable sanity checker. Funding for the RFC Editor function is currently provided by the Internet Society.

7. References

7.1. Normative References

[1] Mockapetris, P., "DOMAIN NAMES - CONCEPTS AND FACILITIES", STD 13, RFC 1034, November 1987.

[2] Mockapetris, P., "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION", STD 13, RFC 1035, November 1987.

[3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[4] Savola, P. and J. Lingard, "Host Threats to Protocol Independent Multicast (PIM)", RFC 5294, August 2008.

7.2. Informative References

[5] Manning, B., "Topology Based Domain Search (TBDS)", Final Report, June 2002, <http://www.dtic.mil/docs/citations/ADA407598>.

[6] Meyer, D., "Administratively Scoped IP Multicast", BCP 23, RFC 2365, July 1998.

Authors' Addresses

Bill Manning
PO 12317
Marina del Rey, CA. 90295
United States

EMail: bmanning@sfc.keio.ac.jp