M. Ohta
Tokyo Institute of Technology

Incremental Zone Transfer in DNS

Status of this Memo

Abstract

This document proposes extensions to the DNS protocols to provide an incremental zone transfer (IXFR) mechanism.

1. Introduction

For rapid propagation of changes to a DNS database [STD13], it is necessary to reduce latency by actively notifying servers of the change. This is accomplished by the NOTIFY extension of the DNS [NOTIFY].

The current full zone transfer mechanism (AXFR) is not an efficient means to propagate changes to a small part of a zone, as it transfers the entire zone file.

Incremental transfer (IXFR) as proposed is a more efficient mechanism, as it transfers only the changed portion(s) of a zone.

In this document, a secondary name server which requests IXFR is called an IXFR client and a primary or secondary name server which responds to the request is called an IXFR server.

The current AXFR specification in [STD13] is very terse, and in practice it does not contain sufficient information to construct interoperable implementations. This memo assumes AXFR protocol used in existing interoperable implementations.

2. Brief Description of the Protocol

If an IXFR client, which likely has an older version of a zone, thinks it needs new information about the zone (typically through SOA refresh timeout or the NOTIFY mechanism), it sends an IXFR message containing the SOA serial number of its, presumably outdated, copy of the zone.

An IXFR server should keep record of the newest version of the zone and the differences between that copy and several older versions. When an IXFR request with an older version number is received, the IXFR server needs to send only the differences required to make that version current. Alternatively, the server may choose to transfer the entire zone just as in a normal full zone transfer.

When a zone has been updated, it should be saved in stable storage before the new version is used to respond to IXFR (or AXFR) queries. Otherwise, if the server crashes, data which is no longer available may have been distributed to secondary servers, which can cause persistent database inconsistencies.

If an IXFR query with the same or newer version number than that of the server is received, it is replied to with a single SOA record of the server's current version, just as a SOA query before TCP AXFR.

Transport of a query may be by either UDP or TCP. If an IXFR query is via UDP, the IXFR server may attempt to reply using UDP if the entire response can be contained in a single UDP packet. If the UDP reply does not fit, the query is responded to with a single SOA record of the server's current version to inform the client that a TCP query should be initiated.

Thus, a client should first make an IXFR query using UDP. If the query type or other part of the query is not recognized by the server, an AXFR (preceded by a UDP SOA query) should be tried, ensuring backward compatibility. If the query response is a single packet with the entire new zone, or if the server does not have a newer version than the client, everything is done. Otherwise, a TCP IXFR query should be tried.

To ensure integrity, servers should use UDP checksums for all UDP responses. A cautious client which receives a UDP packet with a checksum value of zero should ignore the result and try a TCP IXFR instead.

The query type value of IXFR assigned by IANA is 251.

3. Query Format

The IXFR query packet format is the same as that of a normal DNS query, but with the query type being IXFR and the authority section containing the SOA record of client's version of the zone.

4. Response Format

If incremental zone transfer is not available, the entire zone is returned. The first and the last RR of the response is the SOA record of the zone. I.e. the behavior is the same as an AXFR response except the query type is IXFR.

If incremental zone transfer is available, one or more difference sequences is returned. The list of difference sequences is preceded and followed by a copy of the server's current version of the SOA.

Each difference sequence represents one update to the zone (one SOA serial change) consisting of deleted RRs and added RRs. The first RR of the deleted RRs is the older SOA RR and the first RR of the added RRs is the newer SOA RR.

Modification of an RR is performed first by removing the original RR and then adding the modified one.

Each individual difference sequence must leave the zone in a consistent state with contents identical to those visible in the master at the time identified by the new SOA serial number. During a transfer, the slave server may save the zone data to stable storage and use it in responding to queries after applying one or more complete difference sequences even if they do not yet form a complete incremental transfer.

A difference sequence which indicates the removal of a non-existent RR is an indication of an error that the IXFR client is out-of-sync with the IXFR server. The IXFR SHOULD be aborted, and an AXFR requested from the same server. A difference sequence which indicates the addition of a seemingly duplicate (though a node may have multiple TXT RR's with duplicate content) or conflicting RR may just be a malformed zone. In any case the IXFR should be aborted and AXFR performed.

The sequences of differential information are ordered oldest first newest last. Thus, the differential sequences are the history of changes made since the version known by the IXFR client up to the server's current version.

RR sets (RRs of the same RR types) in the incremental transfer messages may be partial. For example, if a single RR of multiple RRs of the same RR type changes, only the changed RR needs to be transferred.

An IXFR client, should only replace an older version with a newer version after all the differences have been successfully processed.

An incremental response is different from that of a non-incremental response in that it begins with two SOA RRs, the server's current SOA followed by the SOA of the client's version which is about to be replaced.

A slave receiving an IXFR response needs to classify it as one of the following four cases:

UDP-overflow	An indication that the transfer will not fit in a UDP packet and should be retried over TCP
up-to-date	An indication that the serial number of the request is current and no transfer is necessary
incremental	An incremental transfer
nonincremental	A full zone transfer

Performing this classification requires some care. For example, UDP-overflow responses differ from UDP up-to-date responses only in the value of the SOA serial number. Also, to distinguish between a nonincremental and an incremental transfer, the slave needs to receive the first two response RRs and check whether the second one is a SOA. If the master chose to transmit each RR in a separate TCP message, this involves waiting for a second TCP response message. On the other hand, in the case of an up-to-date response, the slave must not wait for a second TCP message as doing so would cause it to hang waiting for a message the master will never send. Therefore, the slave must examine the first message and eliminate the possibility that it is a TCP up-to-date response before it attempts to receive a second message.

Slaves must not attempt to classify a response based on incidental information such as the presence or absence of a question section, the QTYPE field of a possible question section, or the number of

response RRs in a TCP response message.

An example algorithm for classifying IXFR responses appears in Appendix A.

5. Purging Strategy

An IXFR server can not be required to hold all previous versions forever and may delete them anytime. In general, there is a trade-off between the size of storage space and the possibility of using IXFR.

Information about older versions should be purged if the total length of an IXFR response would be longer than that of an AXFR response. Given that the purpose of IXFR is to reduce AXFR overhead, this strategy is quite reasonable. The strategy assures that the amount of storage required is at most twice that of the current zone information.

Information older than the SOA expire period should also be purged.

6. Optional Condensation of Multiple Versions

An IXFR server may optionally condense multiple difference sequences into a single difference sequence, thus, dropping information on intermediate versions.

This may be beneficial if a lot of versions, not all of which are useful, are generated. For example, if multiple ftp servers share a single DNS name and the IP address associated with the name is changed once a minute to balance load between the ftp servers, it is not so important to keep track of all the history of changes.

But, this feature may not be so useful if an IXFR client has access to two IXFR servers: A and B, with inconsistent condensation results. The current version of the IXFR client, received from server A, may be unknown to server B. In such a case, server B can not provide incremental data from the unknown version and a full zone transfer is necessary.

Condensation is completely optional. Clients can't detect from the response whether the server has condensed the reply or not.

For interoperability, IXFR servers, including those without the condensation feature, should not flag an error even if it receives a client's IXFR request with a version number known not to exist (which means that the server has versions with version numbers newer and older than, but not equal to, the version number) and should,

instead, attempt to perform a full zone transfer by replying with a single SOA record of the server's current version (UDP case) or with a full zone content (UDP or TCP case).

7. Example

Given the following three generations of data with the current serial number of 3,

```
example.domain.     IN SOA ns.example.domain. rt.example.domain. (
                                1 600 600 3600000 604800)
                    IN NS  ns.example.domain.
ns.example.domain.  IN A   10.0.0.1
ftp.example.domain. IN A   10.0.1.1
```

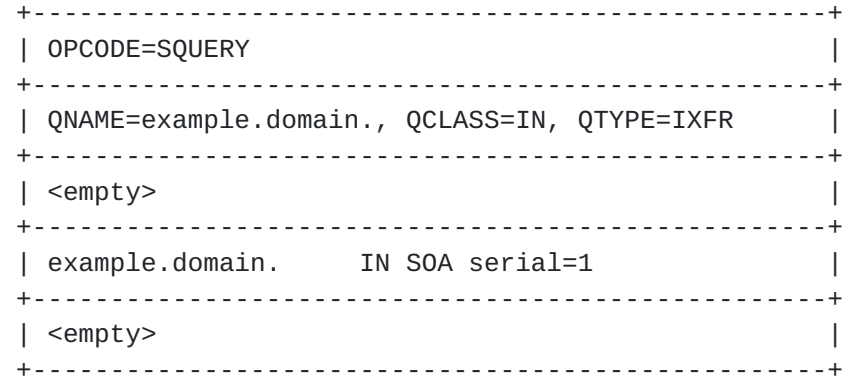
ftp.example.domain. is removed and www.example.domain. is added.

```
example.domain.     IN SOA ns.example.domain. rt.example.domain. (
                                2 600 600 3600000 604800)
                    IN NS  ns.example.domain.
ns.example.domain.  IN A   10.0.0.1
www.example.domain. IN A   10.0.1.2
                    IN A   10.0.2.1
```

One of the IP addresses of www.example.domain. is changed.

```
example.domain.     IN SOA ns.example.domain. rt.example.domain. (
                                3 600 600 3600000 604800)
                    IN NS  ns.example.domain.
ns.example.domain.  IN A   10.0.0.1
www.example.domain. IN A   10.0.3.1
                    IN A   10.0.2.1
```

The following IXFR query



could be replied to with the following full zone transfer message:

```
              +-------------------------------------------------+
   Header     | OPCODE=SQUERY, RESPONSE                         |
              +-------------------------------------------------+
   Question   | QNAME=example.domain., QCLASS=IN, QTYPE=IXFR    |
              +-------------------------------------------------+
   Answer     | example.domain.     IN SOA serial=3             |
              | example.domain.     IN NS  NS.JAIN.AD.JP.       |
              | ns.example.domain.  IN A   10.0.0.1             |
              | www.example.domain. IN A   10.0.3.1             |
              | www.example.domain. IN A   10.0.2.1             |
              | example.domain.     IN SOA serial=3             |
              +-------------------------------------------------+
   Authority  | <empty>                                         |
              +-------------------------------------------------+
   Additional | <empty>                                         |
              +-------------------------------------------------+
```

or with the following incremental message:

```
              +-------------------------------------------------+
   Header     | OPCODE=SQUERY, RESPONSE                         |
              +-------------------------------------------------+
   Question   | QNAME=example.domain., QCLASS=IN, QTYPE=IXFR    |
              +-------------------------------------------------+
   Answer     | example.domain.     IN SOA serial=3             |
              | example.domain.     IN SOA serial=1             |
              | ftp.example.domain. IN A   10.0.1.1             |
              | example.domain.     IN SOA serial=2             |
              | www.example.domain. IN A   10.0.1.2             |
              | www.example.domain. IN A   10.0.2.1             |
              | example.domain.     IN SOA serial=2             |
              | www.example.domain. IN A   10.0.1.2             |
              | example.domain.     IN SOA serial=3             |
              | www.example.domain. IN A   10.0.3.1             |
              | example.domain.     IN SOA serial=3             |
              +-------------------------------------------------+
   Authority  | <empty>                                         |
              +-------------------------------------------------+
   Additional | <empty>                                         |
              +-------------------------------------------------+
```

or with the following condensed incremental message:

```
              +-------------------------------------------------+
   Header     | OPCODE=SQUERY, RESPONSE                         |
              +-------------------------------------------------+
   Question   | QNAME=example.domain., QCLASS=IN, QTYPE=IXFR    |
              +-------------------------------------------------+
```

```
      Answer     | example.domain.     IN SOA serial=3              |
                 | example.domain.     IN SOA serial=1              |
                 | ftp.example.domain. IN A   10.0.1.1              |
                 | example.domain.     IN SOA serial=3              |
                 | www.example.domain. IN A   10.0.3.1              |
                 | www.example.domain. IN A   10.0.2.1              |
                 | example.domain.     IN SOA serial=3              |
                 +--------------------------------------------------+
      Authority  | <empty>                                          |
                 +--------------------------------------------------+
      Additional | <empty>                                          |
                 +--------------------------------------------------+
```

or, if UDP packet overflow occurs, with the following message:

```
                 +--------------------------------------------------+
      Header     | OPCODE=SQUERY, RESPONSE                          |
                 +--------------------------------------------------+
      Question   | QNAME=example.domain., QCLASS=IN, QTYPE=IXFR     |
                 +--------------------------------------------------+
      Answer     | example.domain.     IN SOA serial=3              |
                 +--------------------------------------------------+
      Authority  | <empty>                                          |
                 +--------------------------------------------------+
      Additional | <empty>                                          |
                 +--------------------------------------------------+
```

8. Acknowledgements

The original idea of IXFR was conceived by Anant Kumar, Steve Hotz and Jon Postel.

For the refinement of the protocol and documentation, many people have contributed including, but not limited to, Anant Kumar, Robert Austein, Paul Vixie, Randy Bush, Mark Andrews, Robert Elz, Andreas Gustafsson, Josh Littlefield, Olafur Gudmundsson, William King and the members of the IETF DNSEXT working group.

9. References

[NOTIFY] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC1996, August 1996.

[STD13] Mockapetris, P., "Domain Name System" (RFC1034 and RFC1035), November 1987.

10. Appendix A Appendix A. Pseudocode for response classification

The following pseudocode outlines one possible algorithm for classifying IXFR responses.

```
receive the first response message;
extract the first response RR, always an SOA;
if (the serial number of this SOA RR is less
    than or equal to that of the request) {
    the response is an up-to-date response;
} else {
    if (the response message was received by UDP and
        contains no more RRs after the initial SOA) {
        the response is a UDP-overflow response;
    } else {
        extract the second response RR, waiting for a second TCP
        response message if necessary;
        if (this second RR is an SOA) {
            the response is an incremental transfer;
        } else {
            the response is a nonincremental transfer;
        }
    }
}
```

11. Security Considerations

Though DNS is related to several security problems, no attempt is made to fix them in this document.

This document is believed to introduce no additional security problems to the current DNS protocol.

12. Author's Address

Masataka Ohta
Computer Center, Tokyo Institute of Technology
2-12-1, O-okayama, Meguro-ku, Tokyo 152-8550, JAPAN

Phone: +81-3-5734-3299, Fax: +81-3-5734-3415
EMail: mohta@necom830.hpcl.titech.ac.jp

Comments should be directed to DNSEXT WG <namedroppers@ops.ietf.org>.

13. Full Copyright Statement