SONAR - A Network Proximity Service

draft-moore-sonar-02.txt

1. Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working
documents of the Internet Engineering Task Force (IETF), its areas, and
its working groups. Note that other groups may also distribute working
documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six
months and may be updated, replaced, or obsoleted by other documents at
any time. It is not appropriate to use Internet-Drafts as reference
material or to cite them other than as ‘‘work in progress.’’

To view the entire list of current Internet-Drafts, please check
the "1id-abstracts.txt" listing contained in the Internet-Drafts
Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net
(Northern Europe), ftp.nis.garr.it (Southern Europe), munnari.oz.au
(Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu
(US West Coast).

2. Abstract

SONAR is a service which, when presented with a list of Internet
Protocol addresses, attempts to order that list according to the proxim-
ity from the SONAR server to each address. Given multiple locations for
a network accessible resource, SONAR is designed to help networked
applications make a reasonable choice between those alternatives. This
memo describes the SONAR protocol as well as an experimental implementa-
tion of the SONAR service.

While the name ‘‘SONAR’’ is intended as a pun on the ‘‘ping’’ util-
ity that uses ICMP echo requests to verify network connectivity, the
SONAR service does not specify use of any particular means of estimating
network proximity. To the contrary, the design of SONAR assumes that
the best means of estimating network proximity will change over time and
perhaps from one network location to another. The SONAR protocol is
intended to provide a stable interface for applications that need to
measure network proximity, thus allowing proximity measurement algo-

K. Moore Expires 8 October 1998
3. Introduction

One way to make a network-accessible resource available to a large, and perhaps geographically distributed audience is to provide several replicas (or "mirrors") of that resource at multiple locations. If there are several locations from which a resource can be accessed, and either bandwidth or response-time is important to the user's perceived quality of service, the user will often obtain better service if a "nearby" location of that resource is favored over a "distant" one.

The SONAR service is designed to allow such a choice to be made by an application program. Given several alternative locations from which to access a resource, an application program can query a nearby SONAR server to determine which locations are closer than others. The SONAR server will return, for several of those locations, a metric which provides a rough indication of the proximity of that location to the SONAR server. Using those metrics, the application can then choose a single location from which to access the resource.

SONAR is designed as an alternative to building proximity estimation algorithms into each of several applications. By placing such functionality into a separate network-accessible service, the proximity estimation algorithm can be tuned (in one place) to suit the requirements of a particular site. It also allows SONAR to evolve independently of the applications that use it. When better proximity estimation algorithms are developed, a site or ISP can upgrade its SONAR servers to allow its existing applications to take advantage of those improvements.

SONAR does not guarantee that it finds the "best" location of a resource, nor does it attempt to accurately measure proximity in terms of round-trip-time, hop count, available bandwidth, or any other metric. Instead, the goal of SONAR is to provide a "good" choice without imposing a long delay and without consuming significant network resources in making that choice.

If the time required to access a resource using SONAR is usually less than the time required to access the resource without it, and if the use of SONAR avoids the "worst case" selection, users will see SONAR as beneficial. Likewise, if the amount of network resources consumed by a SONAR server in measuring network proximity is less than the amount of resources that would be consumed when accessing a randomly chosen location, SONAR will improve the utilization of the network.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document, when they appear in all capital letters, are to be interpreted as described in RFC 2119.
4. Protocol

The first version of the SONAR protocol is intentionally as simple as possible while allowing extensibility for future needs.

An application communicates with a SONAR server using UDP port 572. A SONAR query and response each consist of a single UDP datagram. In the descriptions below, all numeric quantities are in standard network byte order (‘‘big-endian’’ format, most significant byte transmitted first).

The format of a SONAR query is as follows:

<table>
<thead>
<tr>
<th>offset</th>
<th>datum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>protocol-version = 0</td>
</tr>
<tr>
<td>4</td>
<td>request-type = 0</td>
</tr>
<tr>
<td>8</td>
<td>request-id</td>
</tr>
<tr>
<td>12</td>
<td>timeout.secs</td>
</tr>
<tr>
<td>16</td>
<td>timeout.usecs</td>
</tr>
<tr>
<td>20</td>
<td>count</td>
</tr>
<tr>
<td>24</td>
<td>address[0]</td>
</tr>
<tr>
<td>28</td>
<td>address[1]</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>24+(count-1)*4</td>
<td>address[count-1]</td>
</tr>
</tbody>
</table>

The protocol-version is used to distinguish this version of the SONAR protocol from other versions. This document describes version 0 of the protocol. A SONAR server MUST check the version-number of each request and return a VERSION_MISMATCH response if that version is not supported.

For SONAR version 0, the only valid request-type is 0, corresponding to a request for proximity information for one or more IP version 4 addresses. It is expected that future versions of SONAR will define additional request types; for example, a query which supports IPv6 addresses and/or allows the user to request a particular kind of...
proximity estimation (RTT vs. bandwidth). This memo documents the format of the SONAR query for request-type 0 only. The format of the portion of the SONAR query following the count field will be different for requests other than type 0.

The request-id is any 4 bytes chosen by the client, and used to associate a response with the particular request that produced. The request-id will be included in the SONAR server’s response.

The timeout is the maximum number of seconds (4 bytes) and microseconds (4 bytes) which the SONAR server should wait before replying. The server MAY return a response sooner than requested. A timeout of zero seconds and zero microseconds requests that the SONAR server return a response immediately, based on whatever proximity data it has at hand.

The count (4 bytes) is the number of addresses for which the SONAR client wants the server to determine relative proximity. The count may be any value between 1 and 185, inclusive. SONAR servers MUST ignore queries with a count of zero, and SHOULD ignore the 186th and later addresses in a query that contains more than 185 addresses.

Following the count are that many Internet Protocol (v4) addresses.

Version 0 of the SONAR service has two possible response formats. A SUCCESS response returned by the SONAR server is in the following format:
Protocol-version will also be zero. For all versions of SONAR, the protocol-version of the response will match that of the request.

(All response-types have their most significant bit set to 1, so that responses can be distinguished from queries.) For a SUCCESS response, response-type will be 0x80000000.

The request-id in the response matches the one from the SONAR query.

The count is the number of IP addresses for which proximity data is being returned. The SONAR server MAY return fewer responses than the number of IP addresses for which proximity information was requested. The SONAR server SHOULD limit the amount of information returned if necessary to prevent the resulting datagram from being fragmented. If a SONAR returns fewer responses than requested, it SHOULD return the proximity estimates of the nearest hosts, discarding the estimates of those that appear to be the most distant. However, a server which has several cached proximity estimates from among those requested MAY return estimates for some subset of the cached estimates without estimating proximity for all hosts requested.
Following the count are several (address, metric) pairs. Each address is an IP address which appeared in the a query, and the metric is an unsigned 4 byte integer which indicates relative proximity of that host to the SONAR server. The addresses returned in a SONAR response are not in any particular order; the client must sort them if needed. The metrics are not necessarily in any particular units, and should be used only for comparison of locations returned within a single response.

A FAILURE response is of the following form:

<table>
<thead>
<tr>
<th>offset</th>
<th>datum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>protocol-version = 0</td>
</tr>
<tr>
<td>4</td>
<td>response-type = 0xc0000000</td>
</tr>
<tr>
<td>8</td>
<td>request-id</td>
</tr>
<tr>
<td>12</td>
<td>low-version</td>
</tr>
<tr>
<td>16</td>
<td>high-version</td>
</tr>
</tbody>
</table>

For a VERSION_MISMATCH response, response-type will be 0xc0000000. (All response-types indicating errors will have their two most significant bits set to 1.)

Low-version is the lowest protocol version supported by the SONAR server.

High-version is the highest protocol version supported by the server.

5. Caveats

Since different SONAR servers may employ different proximity estimation algorithms, metrics obtained from one SONAR server should not be compared with those obtained from a different server. Similarly, metrics obtained from SONAR should not be cached by an application for any significant amount of time.

An application should normally be configured (by whatever means) to use one or more SONAR servers that have a similar view of the network to that application. However, for the case of an application running on a single host which is separated from the rest of the Internet by a slow pipe, where all of the resources of interest are on the ‘‘well-connected’’ side of the link (e.g. a modem dialup connection) the SONAR
server should normally be located on the ‘well-connected’ side.

In general, applications which examine raw IP addresses may be confused by network address translation (NAT) devices. In particular, applications that use SONAR may be confused if the application and SONAR server are on different sides of a NAT box. In the case where a NAT box dynamically maps ‘global’ addresses onto some set of ‘local’ addresses which can be reused (e.g. when mapping global IPv6 addresses onto a local IPv4 network), an application using SONAR will also be confused if the combination of the SONAR server and the application, cache proximity estimation data longer than the NAT box retains its mappings between global and local addresses.

Many sites employ firewalls or filtering in routers to discourage hostile attacks or probes from outside (or inside!) of a perimeter. For example, some sites filter ICMP echo requests. Such sites may interfere with the operation of particular kinds of SONAR servers.

Sites that employ NATs and/or firewalls often utilize application level gateways or proxies. Use of proxies with SONAR is a topic for further study.

6. Finding a SONAR server

SONAR complicates configuration of networked applications, because they need to know the address of one or more SONAR servers to use it. It is therefore desirable to minimize the burden of configuration.

SONAR-aware applications should perform a DNS query of QTYPE=A on the name ‘sonar-server’ to determine if there are any SONAR servers defined for use by the local domain. (This assumes that DNS searches for unqualified names take place relative to one or more local domains.) Even though the local DNS hierarchy does not necessarily correspond to network topology, this approach will work well for many sites.

A future version of SONAR may support location of servers using IP broadcast, IP multicast, DHCP, PPP extensions, service location protocol, or some other means to be determined.

7. Security Considerations

A naive implementation of SONAR could be used as a vehicle for a denial-of-service attack on a network, for instance, by requesting SONAR to probe multicast or broadcast addresses and thus generating large amounts of traffic. For this reason it is recommended that SONAR servers refuse to process queries sent to IP broadcast or multicast addresses, ignore requests to obtain proximity information for broadcast or multicast addresses, and perhaps to ignore requests from remote
The security mechanisms employed by a site to prevent certain kinds of attack may thwart use of SONAR. For example, some sites filter ICMP echo requests from external sites for security reasons, even though other protocols such as ftp or http may be allowed. A SONAR implementation that uses ICMP echo will not be able to estimate proximity to hosts behind such a filter.

8. Author’s address

Keith Moore <moore@cs.utk.edu>

University of Tennessee
107 Ayres Hall
Knoxville, TN 37996-1301
USA

A mailing list has been established to discuss the SONAR protocol and its implementation. To request a subscription to the list, send mail to <sonar-REQUEST@cs.utk.edu>.