MSNL/IP Name Service

Shaw Chuang and Simon Crosby

17th February 1993

1 Introduction

Sensible network administration at the Computer Laboratory requires a mechanism for the resolution of host name to MSNL addresses, similar to the DNS. It was decided to implement a simple MSNL name service for use at the Laboratory which could be used to resolve MSNL address queries. All known MSNL addresses are stored in the name server or in the ANSA trader. The name server queries the ANSA trader to resolve queries for which it does not have the appropriate information. The MSNL name server exports the `name_service` interface, and is assumed to be located at a well known address. It is always bound to port 223 on its host. The name service interface is located in the `master/iface/msdr/name_service.x` interface, and should be consulted for data type definitions.

The current name server supports two naming/address domains, the IP and MSNL worlds. The IP name/address resolution is more useful to a non-Unix host which does not have access to the DNS (or /etc/hosts). On Unix, IP name resolution is not implemented in the name server library.

2 Name Server library

Any Wanda or Unix program which wishes to use MSNL must use the MSNL name service for address resolution if the required MSNL address is not known. The equivalent of the Unix `gethostname, gethostbyname` and `gethostbyaddr` routines are implemented within the MSRPC library on all platforms to support MSNL address resolution. On Unix these routines are named `getmsnlhostname, getmsnlhostbyname` and `getmsnlhostbyaddr`. The mechanism is as follows:

When the appropriate routine is called, the MSRPC library checks to see whether it has an existing session to the MSNL name server. If not then it uses the MSRPC library to become a client of the MSNL name server. It locates the MSNL name server by obtaining its address from the environment variable `MSNL_NAME_SERVERS`. Any number of alternative name servers can be listed in this variable. The format is:

$$\texttt{MSNL\_NAME\_SERVERS} = \text{"}h_{10}.h_{11}.h_{12}.h_{13} : \mathsf{0.0.0.223}, h_{20}.h_{21}.h_{22}.h_{23} : \mathsf{0.0.0.223}\text{"}$$

Where h_{ij} is the j^{th} octet in internet dotted notation of the MSNL address of host h_i. The port `0.0.0.223` is the MSNL name server port.

No assumption should be made on the selection of the current name server from the `MSNL_NAME_SERVERS` environment variable.

Information returned from the name server is returned to the calling process by means of a static area of memory. Results of such calls should thus be copied to ensure they remain valid. On Wanda there is no concurrency protection for this static memory area.

Session bindings to the name server are cached. However, data from the name server is not cached by the library, i.e. every name/address resolution resulted in a call to the name server.

3 Using the Name Server library

For the Wanda library, refer to the man pages gethostname(3) and gethostbyname(3). For the Unix library, man getmsnlhostname(3) and getmsnlhostbyname(3). Even though the current version of the name server gathers the required information by using the internal table and acting as a proxy to the trader and the DNS, the user of the name server library should not make any assumption on this.

4 ANSA Trader (Name Server) Extension

The ANSA trader is mainly used as a database of MSNL addresses and names. The IP part of the name service does not make use of it.

It is possible to have multiple network interfaces to a host, with each network interface having an alias and a network address. The trader name extension for the type MachInfo contains all the aliases and network addresses but it does not provide information as to which alias correspond to which address. The following ANSA trader (name server) extension is made to allow this.

In the context of `/wanda/machines`, a new type, `IFInfo`, has been added and `MachInfo` is compatible with it. Thus the current trader interface type directed acyclic graph (as relevant to us) is as:

MachInfo $\longrightarrow$ IFInfo $\longrightarrow$ wanda

The IFInfo contains properties `Name`, `MSNL`, `Type` and `InterfaceId`. Only the alternative network interface, e.g. `cod-c`, will have an entry in this type. The principle entry, `cod` must always be in the MachInfo type.

When a name to address resolution request (library call of getmsnlhostbyname on Unix or gethostbyname on Wanda) is made (using procedure NS_GET_SAPS_BY_SAPNAME as defined in the NAME_SERVICE interface), the name server performs the following operations:

- Lookup the principle MSNL address which correspond to the name, using a lookup in context `/wanda/machines` with type of `IFInfo` and constraint of property `Name` equal to the name requested.
- If the entry is not in the trader, lookup in a local table. The local table is used to supplement the data in the ANSA trader. This is orginally designed to be a catch-all mechanism to allow entries that do not have entry in the trader. Since it has been decided that both Wanda and Unix machines should have trader entries, this mechanism should not be used other than some exceptional cases.

- If the entry is in the trader, a further lookup to the trader is done to find out about possible additional addresses. The trader lookup the `MSNLAliases` in context `/wanda/machines` with type of `IFInfo`. The constraint is host name in `Aliases`.

The result returned to the user always has the MSNL address of the requested name listed first.

Similarly, when a address to name request (library call of getmsnlhostbyaddr on Unix or gethostbyaddr on Wanda) is made (using procedure NS_GET_SAPNAMES_BY_SAP as defined in the MSRPC interface), the MSNL name server does:

- Lookup the principle interface name that correspond to the address, using a lookup in context `/wanda/machines` with type of `IFInfo` and constraint of property `MSNL` equal to the requested address.
- If the entry is not in the trader, lookup in a local table.
- If the entry is in the trader, do a further lookup to find out the aliases in context `/wanda/machines` with type of `IFInfo`. The constraint is address in `MSNLAliases`.

Again, the result returned to the user always has the name of the requested MSNL address listed first.

4.1 Update the trader

`tco(8)` and `tci(8)` are provided to dump and update the MachInfo part of the trader information. Alternatively naclient can be used if you prefer something more direct.

Standard tco/tci will not be able to show/update the `IFInfo` type. A script `iftco(8)` and `iftci(8)` which provides the functionality of tco and tci respectively for the `IFInfo` interface type.

If you are adding/deleting a network interface from the trader, make sure that you update both `IFInfo` type and the `MSNLAliases` & `Aliases` properties of the `MachInfo` type.

5 Other procedures in the NAME_SERVICE interface

The name server interface definition supports several procedures for accessing name data at the name server. The use of NS_GET_SAPS_BY_SAPNAME and NS_GET_SAPNAMES_BY_SAP has been shown above.

Two additional functions are provided to permit maximum experimental flexibility, namely NS_LOWLEVEL_LOOKUP and NS_GET_SAP_BY_NAME. The former has data types which can be "hacked" at client and server ends to permit arbitrarily marshalled bytes to be sent to the server. The server would have to implement the "msdr-hacked" marshalling routines to understand the data being sent to it.

NS_GET_SAP_BY_NAME permits the client to send a variable length string which will be parsed by the server according to some server-defined "language" to work out what service is required. The return is a set of zero or more saps. Additional routines supply extra information if it is known, to help the server perform the search.

Ideally a client which "knows" what its arguments mean should marshal them for a special rpc call the the server, rather than simply sending a string.

6 Logging

The name server records its activities in a log file. The default location of the log is /tmp/ns.log. Alternative log file name and location can be specified as the first argument to the name server process (see name_service(8)).

The log indicates in which database (internal table or ANSA trader) the name server resolve the query.

7 Wanda world without name server access

i.e. the pre-snap4 world.

You have to maintain you own netdb table !

8 Hostlookup

A simple program that does name server query on the MSNL name/address provided on the argument line. See hostlookup(8).

9 Troubleshooting

When name server lookup returns a null pointer, there are a few possibilities:

- Name/address does not exist

  Blame yourself for giving the server a wrong name/address

- MSNL_NAME_SERVERS is wrong

  Find out if the name service has moved

- Name server is dead

  You can find this out from the Unix errno. Alternatively, hostlookup will tell you this. Get someone to restart the name server !

- Name is not in trader and the local table has not been updated

  You can find this out from the name server log (See section on logging). You are probably using a new Unix host name/address. Just get the name server guru to update the local table, or put it in the trader if is it is appropriate.

- Trader is dead

  You should know by doing a tco. Get the ANSA guru to restart the trader and probably the ANSA world. Restart name server after the trader is up.