Network Working Group
INTERNET DRAFT
Expires: December 31, 1995

Keith Sklower
University of California, Berkeley

Getconninfo(): An alternative to Gethostbyname()
draft-sklower-ipv6-getconninfo-03.txt

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. Internet Drafts 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 a "working draft" or "work in progress."

Please check the 1id-abstracts.txt listing contained in the internet-drafts Shadow Directories to learn the current status of any Internet Draft.

Abstract

This document proposes a uniform programmatic interface to the Internet name resolution system, in which differences between IPv4, IPv6, IPX, and CLNP are irrelevant to the applications programmer. This work was originally motivated by the desire to minimize the impact on standard BSD utilities to support TUBA, but work equally well for IPv6.

Acknowledgements

The author specifically wishes to thank William Durst of the MITRE coporation, Steven Wise of IBM, Michael Karels of BSDI, and Eric Allman of the University of California at Berkeley for many useful discussions of the subject, and thorough review of early versions of the proposal, and especially thank Eric Allman for implementing a prototype. The observation that specifying the pair of name and service would suffice for connecting to a service independent of protocol details was made by Marshall Rose in a proposal to X/Open

for a "Uniform Network Interface".

1. Introduction

Although the IETF normally concerns itself with the details of how information is exchanged between computers, the advent of a revision to the IP protocol has implications for application writers, and a series of comments and suggestions have been made concerning this.

In the present BSD programming environment, and in the internet protocol context, an applications writer must perform a name-to-address and service-to-port translation before issuing socket primitives to initiate a connection to service, or to offer a service. The programmer then combines this information into a protocol specific addressing structure.

Aside from the process of combining the information, it may be possible to code up the rest of the service in a way that makes no other reference IP specific details. Our experience with implementing TCP serivces over CLNP has given us a few cases-in-point.

We give the description of a prototype interface done at Berkeley, which allows an application writer to obtain all the information necessary to establish a connection or offer a service without having to explicitly glue the network address and service parts together.

2. Host to address info

The C-language interface is provided by two routines:

```
#include <netdb.h>

     int
     getconninfo(host, service, clues, results);
     char *host, *service;
     struct conninfo *clues, **results;

     void
     freeconninfo(nuked);
     struct conninfo *nuked;
```

where the conninfo structure is defined below.

This structure contains either the information obtained from the name server, and/or broken-out fields from a line in /etc/hosts, and /etc/services. If the local name server is not running these routines do a lookup in those static files and possibly others.

The getconninfo() function returns a value of 0 (indicating success) or an error code giving the reason for failure. The results paramter is a pointer to a variable which will be overwritten with a pointer to a linked list of objects.

The ``clues'' parameter may be omitted, in which case all possible matches in all address families are returned; or it may be used to limit the queries to return a prespecified set of protocols, with or without aliasing.

If a ``clue'' parameter is provided, it may still be possible to wildcard some of the requirements; a value of AF_UNSPEC for the ci_af element indicates a willing to accept results in any address family, a value of 0 for ci_proto indications a willingness to employ any protocol to provide the service requested, if there is more than one choice in a given family.

The ``host'' parameter may be either a null pointer or a zero length string, meaning to allow connections on any interface in the case of passive opens, or to connect to the the system itself on active opens. This currently works for all commonly implemented protcols. For future protocols where addresses are different depending on whether or not the connections are active or passive, the distinction may be drawn by using the CI_PASSIVE flag.

The ``service'' paramter may also be null or of zero length, if it is desired to use only the address for network management functions such as assigning an address to an interface in the SIOCAIFFADDR ioctl(), or manually to add entires to the routing table.

The conninfo struct is given by:

```
struct conninfo {
     int     ci_flags;
#define CI_PASSIVE      0x1    /* intended for bind() + listen() */
#define CI_CANONICAL    0x2   /* request canonical name */
     int     ci_af;
     int     ci_socktype;
     int     ci_proto;
     int     ci_namelen;
     char    *ci_canon;
     struct  sockaddr *ci_name;
     struct  conninfo *ci_next;
};
```

Members of this structure are:

ci_flags
: This modifies the behavior of getconninfo in ways described above;

ci_af
: The type of address being returned.

ci_socktype
: The socket type required as the second argument in a call to socket().

ci_proto
: The specific protocol required as the third argument in a call to socket().

ci_name
: The binary address, suitable for use as an argument to connect() or bind().

ci_namelen
: The length, in bytes, of the address.

ci_canon
: The canonical name for the host. This is an output of the function, and is only provided if the CI_CANNONICAL flag is set on the clues paramater.

When using the Internet nameserver, getconninfo() would search for the named host in the current domain and its parents unless the name ends in a dot. If the name contains no dot, and if the environment variable ``HOSTALIASES'' contains the name of an alias file, the alias file will first be searched for an alias matching the input name. Getconninfo() will also translate hosts specified in internet dotted-quad notation according to the syntax ``[a.b.c.d]'', and will accept ascii renditions of integers for service names, eg. ``6''.

A non-zero error return can have the following values:

HOST_NOT_FOUND
: No such host is known.

TRY_AGAIN
: This is usually a temporary error and means that the local server did not receive a response from an authoritative server. A retry at some later time may succeed.

NO_RECOVERY
: Some unexpected server failure was encountered. This is a non-

recoverable error.

NO_DATA
The requested name is valid but does not have an IP address; this is not a temporary error. This means that the name is known to the name server but there is no address associated with this name. Another type of request to the name server using this domain name will result in an answer; for example, a mail-forwarder may be registered for this domain.

3. Address to Host Translation

It is sometimes useful to be able to do the reverse translation, i.e. given a binary network address, determine a human readable string or pair of strings which when fed to getconninfo(), would yield the starting result.

The most common use of this is for record keeping purposes, although the mechanism is used in weak verification in SMTP ("you are a charlatan"), and was employed in the Berkeley remote shell services. A reverse name lookup is also done to better effect in the Kerberos code.

We propose a function

```
int
getinfobysockaddr(sa, hostlen, host, servlen, serv)
 struct sockaddr *sa;
 int hostlen, servlen;
 char *host, *serv;
```

As above, a zero value indicates sucess, and a non-zero value gives some indication of the cause of the failure.

The user would allocate space for human readable versions of the hostname and service to be derived from the sockaddr. a length value of zero indicates the application is not interested in having that aspect of the address translated.

Otherwise sufficient space including a trailing zero must be provided for the translation.

4. Author's Address

Keith Sklower
Computer Science Department
380 Soda Hall, MS 1776
University of California
Berkeley, CA 94720-1776

Phone: (510) 642-9587
E-mail: sklower@CS.Berkeley.EDU

5. Expiration Date of this Draft

December 31st, 1995