INTERNET-DRAFT
Expires: August, 2003

Erik Nordmark
Samita Chakrabarti
Sun Microsystems, Inc.
Julien Laganier
Sun Microsystems, Inc.
LIP / ENS-Lyon
February, 2003

IPv6 Socket API for source address selection
draft-chakrabarti-ipv6-addrselect-api-00.txt

Status of this Memo

This Internet Draft expires August, 2003.

Copyright Notice

Abstract

The IPv6 default address selection document describes the rules for selecting default source address by the system and indicates that the applications should be able to reverse the sense of system preference of source address selection for that application through possible API extensions. However, no such socket APIs exist in the basic or advanced IPv6 socket API documents. Hence this document specifies socket level options to prefer a particular source address as per the choice of the applications. It also discusses implications on the name-to-address translation API that performs part of the default address selection. The socket APIs described in this document will be particularly useful for Mobile IPv6 enabled applications and other IPv6 applications which want to choose between temporary and public addresses, CGA (cryptographically generated addresses) and non-CGA addresses etc..

Table of Contents

1. Introduction

This document defines socket extensions to support the non-default choice of source address by the applications. The IPv6 default address selection [1] document has specified the rules for system default source address selection for an outbound IPv6 packet. Privacy considerations [6] have introduced "public" and "temporary" addresses. IPv6 Mobility [3] introduces "home address" and "care-of-address" definitions in the mobile systems. Although it is desirable to have default algorithms for the system to choose the source address of the outgoing IPv6 packet, an application may want to reverse that rule for efficiency and other application specific reasons. Currently IPv6 socket API extensions does not provide a mechanism to choose a particular source address other than simple bind() operation. The bind() operation allows an application to specify a particular source address. Thus in order to use bind() the application itself must make sure that the source address is appropriate for the destination address (e.g., with respect to the interface used to send packets to the destination). The application also needs to make sure about the appropriate scope of source address with respect to the destination address and so on. The mechanism presented in this document allows the application to specify attributes of the source addresses it prefers while still having the system do the rest of the default address selection.

A socket option has been deemed useful for this purpose, as it enables an application ability to make a choice of source address at per-socket basis as well as it can provide flexibility of enabling and disabling choice of source addresses in non-connected sockets. The socket option uses a set of flags for source address preferences. Since source address selection and destination address ordering need to be partially implemented in getaddrinfo() [2] the corresponding set of flags are also defined for that routine.

Thus this document introduces different flags for source address selection that can be used by the applications for Mobility [3], Privacy Extension [6] and CGA [7] scenarios. In future, more flags can be added to designate a choice for a certain type of source address as the needs may arise.

The approach in this document is to allow the application to specify preferences on source addresses and not to be able to specify hard requirements. Thus for instance an application can specify that it prefers temporary addresses but if no temporary addresses are available to the default address selection algorithm, a public address would be chosen instead.

Furthermore, the approach is to define two flags for each purpose, so that an application can specify either that it prefers 'X' or prefers 'not X', or it can choose not to set either of the flags relating to 'X' and leave it up to the system default, perhaps while specifing its preferences for some other attribute of the source addresses.

2. Example Usages

The examples of usages discussed here are limited to applications supporting Mobile IPv6, IPv6 Privacy Extensions and Cryptographically Generated Addresses. Address selection document [1] recommends that home addresses should be preferred over care-of-address when both are configured. However, a mobile node may want to prefer care-of-address as source address for DNS query in the foreign network as it normally means a shorter and local return path compared to the route via the mobile node's home-agent when the query contains home-address as source address. Another example is IKE application which requires care-of-address as its source address for the initial security association pair with Home Agent [3] while the mobile node boots up at the foreign network and wants to do the key exchange before a successful home-registration. Also a Mobile IPv6 aware application may want to toggle between home-address and care-of-address depending on its location and state of the application. It may also want to open different sockets and use home-address as source address for one socket and care-of-address for the others.

In a non-mobile environment, similarly an application may prefer to use temporary address as source address for certain cases. By default, the source address selction rule selects "public" address when both are available. For example, an application supporting web browser and mail-server may want to use "temporary" address for the former and "public" address for the mail-server as a mail-server may require reverse path for DNS records for anti-spam rules.

Similarly, a node may be configured to use the cryptographically genenerated addresses by default, but an application may prefer not to use it. For instance, fping, a debugging tool which tests basic reachability of multiple destinations by sending packets in parallel, may find that the cost and time incurred in proof-of-ownership by CGA verification is not justified.
On the other hand, when a node is not configured for CGA as default, an application may prefer using CGA by setting the socket option. It may subsequently verify that it is truly bound to a CGA by first

calling getsockname() and then recomputing the CGA using the public key of the node.

3. Changes to the Socket Interface

IPv6 Basic API [2] defines socket options for IPv6. This document adds a new socket option at the IPPROTO_IPV6 level. This socket option is called IPV6_SRC_PREFERENCES. It can be used with setsockopt() and getsockopt() calls. This socket option takes a 32bit unsigned integer argument. The argument consists of a number of flags which indicate the choice of source address selection.

The flags defined in this document are:

```
IPV6_PREFER_SRC_HOME
IPV6_PREFER_SRC_COA
IPV6_PREFER_SRC_TMP
IPV6_PREFER_SRC_PUBLIC
IPV6_PREFER_SRC_CGA
IPV6_PREFER_SRC_NONCGA
```

The following example illustrates how it is used:

```
uint32_t flags = IPV6_PREFER_SRC_COA;

if (setsockopt(s, IPPROTO_IPV6, IPV6_SRC_PREFERENCES,
        (char *) &flags, sizeof (flags)) == -1) {
       perror("setsockopt IPV6_SRC_REFERENCES");
}
```

When the IPV6_SRC_PREFERENCES is successfully set with setsockopt(), the option value given is used to specify source address for any connection initiation through the socket and all subsequent packets sent via that socket. If the option is not set, the system selects a default value. Setting conflicting flags at the same time results in the error EINVAL.

It is recommended that the application does a getsockopt() prior calling to setsockopt() call so that it can save the existing source address preference value, in the cases when the application might need to restore the preferences.

The constants mentioned in this section are defined in <netinet/in.h>.

4. Changes to the protocol-independent nodename translation

Section 8 of Default Address Selction [1] document indicates about possible implementation strategy for getaddrinfo() [2]. getaddrinfo() collects available source addresses from the network layer and then it sorts the list of source addresses as per source address selection rules. Thus if an application sets setsockopt() IPV6_SRC_PREFERENCES option to alter the default address selection rules , it must make sure that it calls getaddrinfo() with the corresponding flags specified in this section. This will ensure correct behavior of getaddrinfo() destination address selection based on the sorted list of source addresses as per the socket source address selection preferences.

The following flags are added for the ai_flags in addrinfo data structure defined in Basic IPv6 Socket API Extension [2].

```
AI_PREFER_SRC_HOME
AI_PREFER_SRC_COA
AI_PREFER_SRC_TMP
AI_PREFER_SRC_PUBLIC
AI_PREFER_SRC_CGA
AI_PREFER_SRC_NONCGA
```

The above flags are ignored for the AF_INET address family. If a returned address is an IPv4 address (either as AF_INET6 when AI_V4MAPPED, or as AF_INET) then the source preference flags have no effect.

If conflicting flags such as AI_PREFER_SRC_HOME and AI_PREFER_SRC_ COA are set, the getaddrinfo() fails with an error EAI_BADFLAGS[2].

Some valid sequences of flags would be:

```
AI_PREFER_SRC_HOME | AI_PREFER_SRC_PUBLIC
AI_PREFER_SRC_COA  | AI_PREFER_SRC_PUBLIC
AI_PREFER_SRC_HOME | AI_PREFER_SRC_CGA
AI_PREFER_SRC_HOME | AI_PREFER_SRC_NONCGA
AI_PREFER_SRC_COA  | AI_PREFER_SRC_CGA
AI_PREFER_SRC_COA  | AI_PREFER_SRC_NONCGA
```

All the constants mentioned in this section for ai_flags are defined in <netdb.h>.

5. IPv4-Mapped IPv6 Addresses

IPv4-Mapped IPv6 addresses are not supported for setting preference on home, care-of-address, CGA, non-CGA, public or privacy auto-configured addresses as source addresses. Because they are all pure IPv6 addresses.

6. Security Considerations

This document conforms to the same security implications as specified in IPv6 Basic Socket API [2] document. It is also recommended that the applications set IPV6_V6ONLY IP level socket option to permit the nodes to not process IPv4 packets as IPv4 Mapped addresses. Allowing applications to specify a preference for temporary addresses provides per-application (and per-socket) ability to use the privacy benefits of the temporary addresses.

7. Open Issues

- Are there more flags we should define at this point in time? For instance, PREFER_LARGEST_SCOPE?

- Is there a need for REQUIRE flags in addition to or instead of the PREFER flags? Note that in general it isn't possible to verify that a requirement can be satisfied until sendto() or connect() (when the destination address is known) thus this would result in late errors being reported to the application.

- Is there a need for "validation" functions to go with these preferences such as functions that check whether an address is a temporary address?

8. References

Normative references:

[1] Richard Draves, "Default Address Selection for IPv6", draft-ietf-ipv6-default-addr-select-09.txt, August 6, 2002.

[2] R.E. Gilligan, S. Thomson, J. Bound, J. McCann, W. R. Stevens, "Basic Socket Interface Extensions for IPv6", draft-ietf-ipngwg-rfc2553bis-10.txt, December, 2002.

Informative references:

[3] Johnson, D., Perkins, C., Arkko, J., "Mobility Support in IPv6" draft-ietf-mobileip-ipv6-20.txt, January, 2003.

[4] Deering, S., Hinden, R., "Internet Protocol, Version 6 (IPv6), Specification", RFC 2460, Dec. 1998.

[5] Stevens, W. R, Thomas, M., Nordmark, E., Jinmei, T., "Advanced Sockets API for IPv6", draft-ietf-ipngwg-rfc2292bis-07.txt April 19, 2002.

[6] Narten, T. and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 3041, January 2001.

[7] Montenegro, G. and C. Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses.", NDSS 2002, February 2002.

[8] Castelluccia, C. and G. Montenegro, "Securing Group Management in IPv6 with Cryptographically Generated Addresses", draft-irtf-gsec-sgmv6-01 (work in progress), July 2002.

9. Acknowledgements

The authors like to thank members of mobile-ip and ipv6 working groups for useful discussion on this topic. Richard Draves and Dave Thaler suggested that getaddrinfo also needs to be considered along with the new socket option. Gabriel Montenegro suggested that CGAs may also be considered in this document. Thanks to Alain Durand, Renee Danson, Alper Yegin and Francis Dupont for useful discussions.

10. Authors' Addresses

Erik Nordmark
Sun Microsystems Laboratories, Europe
180 Avenue de l'Europe
38334 Saint Ismier, France
Email: Erik.Nordmark@sun.com

Samita Chakrabarti
Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054, USA
Email: samita.chakrabarti@Sun.com

Julien Laganier
Sun Microsystems Laboratories, Europe
180 Avenue de l'Europe
38334 Saint Ismier, France
Email: Julien.Laganier@Sun.COM