Distributed Mobility Management (DMM) J. Korhonen
Internet-Draft Nokia Siemens Networks
Updates: 4861 (if approved) B. Patil
Intended status: Standards Track Nokia
Expires: August 26, 2012 S. Gundavelli
Cisco
February 23, 2012

IPv6 Prefix Mobility Management Properties
draft-korhonen-dmm-prefix-properties-00.txt

Abstract

This specification defines an extension to the IPv6 Neighbor Discovery protocol and its Prefix Information Option. The Prefix Information Option is extended with flag bits that describe the mobility management properties associated to the prefix. This specification updates RFC4861.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Table of Contents

1. Introduction

This specification defines an extension to the IPv6 Neighbor Discovery protocol and its Prefix Information Option (PIO) [RFC4861]. The Prefix Information Option is extended with flag bits that describe the mobility management properties associated to the prefix, and at the same time defines corresponding source address selection hint flags to the IPv6 Socket API for Source Address Selection [RFC5014].

The IPv6 Socket API for Source Address Selection [RFC5014] already covers Mobile IPv6 [RFC6275] and allows selecting between a home address (HoA) and a care-of address (CoA). A mobile node (MN) with a client based mobility IP stack is supposed to know which prefixes are CoA(s) and/or HoA(s). The extensions to [RFC4861] are minimal in a sense that they do not define new functionality to any existing mobility protocol but instead add an explicit indication of network based mobility knowledge into the IPv6 stateless address autoconfiguration (SLAAC). This would allow for network based mobility solutions, such as Proxy Mobile IPv6 [RFC5213] or GTP [TS.29274] to explicitly indicate that their prefixes have mobility, and therefore, the MN IP stack can make an educated selection between prefixes that have mobility and those that do not. There is also a potential need to extend both [RFC3493] and [RFC5014] in order to provide required hooks into socket APIs.

The underlying assumption is that a MN has multiple prefixes to choose from. Typically this means either the MN has multiple interfaces or an interface has been configured with multiple prefixes. This specification does not make a distinction between these alternatives and does not either make any assumptions how the possible transfer of a prefix is done between interfaces in the case a network based mobility solution is used.

2. Background and Motivation

[Discussion: explain the background and subsequently the motivation, which lead to "coloring" prefixes, and what we expect to gain from this extension to IPv6 addressing & neighbor discovery protocol. To be written later.]

3. Option Formats

Neighbor Discovery messages include zero or more options, some of which may appear multiple times in the same message. Options should be padded when necessary to ensure that they end on their natural 64-

bit boundaries. Figure 1 illustrates a Prefix Information Option [RFC4861] that is extended with flag bits describing the mobility properties of the prefix:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       3       |       4       | Prefix Length |L|A| C | Rsvd1 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Valid Lifetime                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Preferred Lifetime                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Reserved2                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                            Prefix                             +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Extended Prefix Information Option

'C' 2-bit flag field describing the mobility properties of the prefix. The following properties are defined:

00 No specific property associated to the prefix. The prefix is treated according to RFC4861.

01 The prefix provides network based mobility and will remain unchanged at the valid lifetime of the prefix.

10 The prefix provides network based mobility but only within a limited area, thus the end host must be prepared that the prefix may become invalid before the valid or even the preferred lifetime expire.

11 Reserved. Treated as '00' by the receiver.

A common use case is to define 'C' flags when the 'A'=1 i.e. when Stateless Address AutoConfiguration (SLAAC) is used. However, it is possible to associate 'C' flags also to prefixes when 'A'=0. In cases when there are multiple learned prefixes with 'C' flags set to a non-zero value that can also be aggregated, then the longest prefix takes precedence.

4. Host Considerations

4.1. Internal Data Structures

The host internal data structures need to be extended with 'mobility property' flag information associated to the learned prefix and configured addresses. How this is accomplished is host implementation specific. It is also a host implementation issue how an application can learn or query mobility properties of an address or a prefix. One possibility is to provide such information through the socket API extensions (see discussion in Appendix A). Other possibilities include the use of e.g., ioctl() or NetLink [RFC3549] extensions.

4.2. Default Address Selection

The 'mobility property' flags are only used as a hint. They do not affect the existing [RFC3484] automatically. A specific rule to host's policy table has to be inserted by an application or some daemon process. Alternatively, an application can express its address mobility property preferences through the socket API extensions (see discussion in Appendix A), which means the socket library or middleware has to modify [RFC3484] policy table or algorithm.

5. Security Considerations

Existing Prefix Information Option related security considerations apply as described in [RFC4861] and [RFC4191]. A malicious node on the shared link could include such 'mobility property' flags in a Prefix Information Option causing the host to learn wrong information regarding the prefix and thus make misguided selection of prefixes on the link. Similarly a malicious middleman on the link could modify 'mobility property' flags in a Prefix Information Option causing misguided selection of prefixes. In order to avoid on-link attacks, SeND [RFC3971] can be used to reject Router Advertisements from potentially malicious nodes and guarantee integrity protection of the Router Advertisements.

6. IANA Considerations

Section 3 defines a new flag bits (2 bit 'C' flag) to the IPv6 Neighbor Discovery protocol's Prefix Information Option [RFC4861].

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

7.2. Informative References

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[RFC3549] Salim, J., Khosravi, H., Kleen, A., and A. Kuznetsov, "Linux Netlink as an IP Services Protocol", RFC 3549, July 2003.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

[RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, September 2007.

[RFC5213] Gundavelli, S., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, August 2008.

[RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

[TS.29274]
3GPP, "3GPP Evolved Packet System (EPS); Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C)", 3GPP TS 29.060 8.11.0, December 2010.

Appendix A. Additions to the Socket Interface and the Protocol-Independent Nodename Translation

Following sections are for informational and discussion purposes only.

This specification also describes non-normative extensions to both Socket Interface [RFC3493][RFC5014] and the Protocol-Independent Nodename Translation [RFC5014]. These socket APIs and DNS resolver APIs extension correspond to the Prefix Information option mobility properties flag bit settings.

A.1. Socket Interface

This specification extends the socket option IPV6_ADDR_PREFERENCES at the IPPROTO_IPV6 level. The following new flags are defined to query, alter or set the default rule of source address selection rules [RFC3484]. They are also defined as a result of including the <netinet/in.h> header:

```
IPV6_PREFER_SRC_HNP     /* Prefer Home Network Prefix derived IPv6
                           address as source */

IPV6_PREFER_SRC_HNP_TMP /* Prefer temoporary Home Network Prefix
                           derived IPv6 address as source */
```

A.2. Protocol-Independent Nodename Translation

the Default Address Selection [RFC3484] document indicates possible implementation strategies for getaddrinfo(). The address selection hint flags for the getaddrinfo() specificed in this document extend the 'int ai_eflags' field in the struct addrinfo [RFC5014][RFC3493].

The IPV6 source address preference values (IPV6_PREFER_SRC_HNP and IPV6_PREFER_SRC_HNP_TMP) defined for the IPV6_ADDR_PREFERENCES socket option are also defined as address selection preference flags in <netdb.h> header for the "ai_eflags" extended flag-set field of the addrinfo data structure.

Similarly to [RFC5014], if contradictory flags, such as IPV6_PREFER_SRC_HOME and IPV6_PREFER_SRC_HNP*, are set in ai_eflags, the getaddrinfo() fails and returns the value EAI_BADEXTFLAGS. This error value MUST be interpreted into a descriptive text string when passed to the gai_strerror() function [RFC3493].

Authors' Addresses

Jouni Korhonen
Nokia Siemens Networks
Linnoitustie 6
FIN-02600 Espoo
Finland

Email: jouni.nospam@gmail.com

Basavaraj Patil
Nokia
6021 Connection Drive
Irving, TX 75039
USA

Email: basavaraj.patil@nokia.com

Sri Gundavelli
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sgundave@cisco.com