An Extension of Format for IPv6 Scoped Addresses

<draft-ietf-ipngwg-scopedaddr-format-02.txt>

Status of this Memo

Abstract

This document defines an extension of the format for IPv6 scoped addresses. In the format, a zone identifier is attached to a scoped address in order to supplement ambiguity of the semantics of the address. Using the format with some library routines will make scope-aware applications simpler.

1. Introduction

There are several types of scoped addresses defined in the "IPv6 Addressing Architecture" [ADDRARCH]. Since uniqueness of a scoped address is guaranteed only within a corresponding zone [SCOPEARCH], the semantics for a scoped address is ambiguous on a zone boundary. For example, when a user specifies to send a packet from a node to a link-local address of another node, the user must specify the link of the destination as well, if the node is attached to more than one link.

This characteristic of scoped addresses may introduce additional cost to scope-aware applications; a scope-aware application may have to provide a way to specify a scope zone for each scoped address (e.g. a specific link for a link-local address) that the application uses. Also, it is hard for a user to "cut and paste" a scoped address due to the ambiguity of its scope.

Applications that are supposed to be used in end hosts like telnet, ftp, and ssh, are not usually aware of scoped addresses, especially of link-local addresses. However, an expert user (e.g. a network administrator) sometimes has to give even link-local addresses to such applications.

Here is a concrete example. Consider a multi-linked router, called "R1", that has at least two point-to-point interfaces. Each of the interfaces is connected to another router, called "R2" and "R3". Also assume that the point-to-point interfaces are "unnumbered", that is, they have link-local addresses only.

Now suppose that the routing system on R2 hangs up and has to be reinvoked. In this situation, we may not be able to use a global address of R2, because this is a routing trouble and we cannot expect that we have enough routes for global reachability to R2.

Hence, we have to login R1 first, and then try to login R2 using link-local addresses. In such a case, we have to give the link-local address of R2 to, for example, telnet. Here we assume the address is fe80::2.

Note that we cannot just type like
% telnet fe80::2
here, since R1 has more than one interface (i.e. link) and hence the telnet command cannot detect which link it should try to connect.

Although R1 could spray neighbor solicitations for fe80::2 on all links that R1 attaches in order to detect an appropriate link, we cannot completely rely on the result. This is because R3 might also assign fe80::2 to its point-to-point interface and might return a neighbor advertisement faster than R2. There is currently no mechanism to (automatically) resolve such conflict. Even if we had one, the administrator of R3 might not accept to change the link-local address especially when R3 belongs to a different organization from R1's.

Another example is an EBGP peering. When two IPv6 ISPs establish an EBGP peering, using a particular ISP's global addresses for the peer would be unfair, and using their link-local addresses would be better in a neutral IX. In such a case, link-local addresses should be specified in a router's configuration file and the link for the addresses should be disambiguated, since a router usually connects to multiple links.

This document defines an extension of the format for scoped addresses in order to overcome this inconvenience. Using the extended format with some appropriate library routines will make scope-aware applications simpler.

2. Assumptions and Definitions

In this document we adopt the same assumption of characteristics of scopes as described in the scoped routing document [SCOPEDROUTING].

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, if and where they appear in this document, are to be interpreted as described in [KEYWORDS].

3. Proposal

The proposed format for scoped addresses is as follows:

```
<scoped_address>%<zone_id>
```

where
<scoped_address> is a literal IPv6 address,
<zone_id> is a string to identify the scope zone of the address, and
`%' is a delimiter character to distinguish between
<scoped_address> and <zone_id>.

The following subsections describe detail definitions and concrete examples of the format.

3.1 Scoped Addresses

The proposed format is applied to all kinds of unicast and multicast scoped addresses, that is, all non-global unicast and multicast addresses.

The format should not be used for global addresses. However, an implementation which handles addresses (e.g. name to address mapping functions) MAY allow users to use such a notation (see also Appendix C).

3.2 Zone Identifiers

An implementation SHOULD support at least numerical identifiers as <zone_id>, which are non-negative decimal numbers. Positive identifiers MUST uniquely specify a single zone for a given scoped address. An implementation MAY use zero to have a special meaning, for example, a meaning that no scope zone is specified.

An implementation MAY support other kinds of non-null strings as <zone_id> unless the strings conflict with the delimiter character. The precise semantics of such additional strings is implementation dependent.

One possible candidate of such strings would be interface names, since interfaces uniquely disambiguate any type of scopes [SCOPEDROUTING]. In particular, if an implementation can assume that there is a one-to-one mapping between links and interfaces (and the assumption is usually reasonable,) using interface names as link identifiers would be natural.

An implementation could also use interface names as <zone_id> for

larger scopes than links, but there might be some confusion in such use. For example, when more than one interface belongs to a same site, a user would be confused about which interface should be used. Also, a mapping function from an address to a name would encounter a same kind of problem when it prints a scoped address with an interface name as a zone identifier. This document does not specify how these cases should be treated and leaves it implementation dependent.

It cannot be assumed that a same identifier is common to all nodes in a zone. Hence, the proposed format MUST be used only within a node and MUST NOT be sent on a wire.

3.3 Examples

Here are examples. The following addresses

```
fe80::1234 (whose link identifier is 1)
fec0::5678 (whose site identifier is 2)
```

    ff02::9abc (whose link identifier is 5)
    ff08::def0 (whose organization identifier is 10)

would be represented as follows:

    fe80::1234%1
    fec0::5678%2
    ff02::9abc%5
    ff08::def0%10

If we use interface names as <zone_id>, those addresses could also be represented as follows:

    fe80::1234%ne0
    fec0::5678%ether2
    ff02::9abc%pvc1.3
    ff08::def0%interface10

where the interface "ne0" belongs to link 1, "ether2" belongs to site 2, and so on.

3.4 Omitting Zone Identifiers

This document does not intend to invalidate the original format for scoped addresses, that is, the format without the scope identifier portion. An implementation SHOULD rather provide a user with a "default" zone of each scope and allow the user to omit zone identifiers.

Also, when an implementation can assume that there is no ambiguity of any type of scopes on a node, it MAY even omit the whole functionality to handle the proposed format. An end host with a single interface would be an example of such a case.

4. Combinations of Delimiter Characters

There are other kinds of delimiter characters defined for IPv6 addresses. In this section, we describe how they should be combined with the proposed format for scoped addresses.

The IPv6 addressing architecture [ADDRARCH] also defines the syntax of IPv6 prefixes. If the address portion of a prefix is scoped one and the scope should be disambiguated, the address portion SHOULD be in the proposed format. For example, the prefix fec0:0:0:1::/64 on a site whose identifier is 2 should be represented as follows:

```
fec0:0:0:1::%2/64
```

In this combination, it is important to place the zone identifier portion before the prefix length, when we consider parsing the format by a name-to-address library function (see Appendix A). That is, we can first separate the address with the zone identifier from the prefix length, and just pass the former to the library function.

The preferred format for literal IPv6 addresses in URL's are also defined [URLFORMAT]. When a user types the preferred format for an IPv6 scoped address whose zone should be explicitly specified, the user could use the proposed format combined with the preferred format.

However, the typed URL is often sent on a wire, and it would cause confusion if an application did not strip the <zone_id> portion before sending. Also, the proposed format might conflict with the URI syntax [URI], since the syntax defines the delimiter chracter (`%') as the escape character.

Hence, this document does not specify how the proposed format should be combined with the preferred format for literal IPv6 addresses. As for the conflict issue with the URI format, it would be better to wait until the relationship between the preferred format and the URI syntax is clarified. Actually, the preferred format for IPv6 literal addresses itself has same kind of conflict. In any case, it is recommended to use an FQDN instead of a literal IPv6 address in a URL, whenever an FQDN is available.

5. Related Issues

In this document, it is assumed that a zone identifier is not necessarily common in a single zone. However, it would be useful if a common notation is introduced (e.g. an organization name for a site). In such a case, the proposed format could be commonly used to designate a single interface (or a set of interfaces for a multicast address) in a scope zone.

When the network configuration of a node changes, the change may affect <zone_id>. Suppose that the case where numerical identifiers are sequentially used as <zone_id>. When a network interface card is newly inserted in the node, some identifiers may have to be renumbered accordingly. This would be inconvenient, especially when addresses with the numerical identifiers are stored in non-volatile storage and reused after rebooting.

6. Security Considerations

Since the use of this approach to represent IPv6 scoped addresses is restricted within a single node, it does not cause a security atack from the outside of the node.

However, a malicious node might send a packet that contains a textual IPv6 scoped address in the proposed format, intending to deceive the receiving node about the zone of the scoped address. Thus, an implementation should be careful when it receives packets that contain IPv6 scoped addresses as data.

Appendix A. Interaction with API

The proposed format would be useful with some library functions defined in the "Basic Socket API" [BASICAPI], the functions which translate a nodename to an address, or vice versa.

For example, if getaddrinfo() parses a literal IPv6 address in the proposed format and fills an identifier according to <zone_id> in the sin6_scope_id field of a sockaddr_in6 structure, then an application would be able to just call getaddrinfo() and would not have to care about scopes.

Also, if getnameinfo() returns IPv6 scoped addresses in the proposed format, a user or an application would be able to reuse the result by a simple "cut and paste" method.

The kernel should interpret the sin6_scope_id field properly in order to make these extensions feasible. For example, if an application passes a sockaddr_in6 structure that has a non-zero sin6_scope_id value to the sendto() system call, the kernel should send the packet to the appropriate zone according to the sin6_scope_id field. Similarly, when a packet arrives from a scoped address, the kernel should detect the correct zone identifier based on the address and the receiving interface, fill the identifier in the sin6_scope_id field of a sockaddr_in6 structure, and then pass the packet to an application via the recvfrom() system call, etc.

Note that the ipng working group is now revising the basic socket API in order to support scoped addresses appropriately. When the revised version is available, it should be preferred to the description of this section.

Appendix B. Implementation Experiences

The WIDE KAME IPv6 stack implements the extension to the getaddrinfo() and the getnameinfo() functions described in Appendix A of this document. The source code is available as free software, bundled in the KAME IPv6 stack kit.

The current implementation assumes a one-to-one mapping between links and interfaces, and hence it uses interface names as <zone_id> for links.

For instance, the implementation shows its routing table as follows:

```
Internet6:
Destination       Gateway                        Flags  Intface
default           fe80::fe32:93d1%ef0            UG      ef0
```

This means that the default router is fe80::fe32:93d1 on the link identified by the interface "ef0". A user can "cut and paste" the result in order to telnet to the default router like this:

```
% telnet fe80::fe32:93d1%ef0
```

even on a multi-linked node.

As another example, we show how the implementation can be used for the problem described in Section 1.

We first confirm the link-local address assigned to the point-to-point interface of R2:

```
(on R1)% ping ff02::1%pvc0

PING(56=40+8+8 bytes) fe80::1 --> ff02::1
16 bytes from fe80::1%lo0, icmp_seq=0 hlim=64 time=0.474 ms
16 bytes from fe80::2%pvc0, icmp_seq=0 hlim=64 time=0.374 ms(DUP!)
...
(we assume here that the name of the point-to-point interface
on R1 toward R2 is "pvc0" and that the link-local address on
the interface is "fe80::1".)
```

So the address should be fe80::2. Then we can login R2 using the address by the telnet command without ambiguity:

```
% telnet fe80::2%pvc0
```

Though the implementation supports the extended format for all type of scoped addresses, our current experience is limited to link-local addresses. For other type of scopes, we need more experience.

The implementation also supports the notion of "default" scope zone as described in Section 3.4. If a user specified "pvc0" as the default link in the above example, the user can just type

```
% telnet fe80::2
```

then the kernel will automatically use the link identified by "pvc0" as the link of the address fe80::2.

Appendix C. A Comprehensive Description of KAME's getXXXinfo Functions

The following tables describe the behavior of the KAME's implementation we mentioned in Appendix B using concrete examples. Note that those tables are not intended to be standard specifications of the extensions but are references for other implementors.

Those tables summarize what value the getXXXinfo functions return against various arguments. For each of two functions we first explain typical cases and then show non-typical ones.

The tables for getaddrinfo() have four columns. The first two are arguments for the function, and the last two are the results. The tables for getnameinfo() also have four columns. The first three are arguments, and the last one is the results.

Columns "Hostname" contain strings that are numeric or non-numeric IPv6 hostnames.

Columns "NI_NUMERICHOST" or "NHOST" show if the NI_NUMERICHOST is set to flags for the corresponding getXXXinfo function. Columns "NSCOPE" show if the NI_NUMERICSCOPE is set to flags for the corresponding getnameinfo function. The value "1" means the flag is set, and "0" means the flag is clear. "-" means that the field does not affect the result.

Columns "sin6_addr" contain IPv6 binary addresses in the textual format, which mean the values of the sin6_addr field of the corresponding sockaddr_in6 structure.

Columns "sin6_scope_id" contain numeric numbers, which mean the values of the sin6_scope_id field of the corresponding sockaddr_in6 structure.

If necessary, we use an additional column titled "N/B" to note something special.

If an entry of a result column has the value "Error", it means the corresponding function fails.

In the examples, we assume the followings:

- The hostname "foo.kame.net" has a AAAA DNS record "3ffe:501::1". We also assume the reverse map is configured correctly.
- There is no FQDN representation for scoped addresses.
- The numeric link identifier for the interface "ne0" is 5.
- We have an interface belonging to a site whose numeric identifier is 10.
- The numeric identifier "20" is invalid for any type of scopes.
- We use the string "none" as an invalid non-numeric zone identifier.

Typical cases for getaddrinfo():

Hostname	NI_NUMERICHOST	sin6_addr	sin6_scope_id
"foo.kame.net"	0	3ffe:501::1	0
"3ffe:501::1"	-	3ffe:501::1	0
"fec0::1%10"	-	fec0::1	10
"fe80::1%ne0"	-	fe80::1	5
"fe80::1%5"	-	fe80::1	5

Typical cases for getnameinfo():

sin6_addr	sin6_scope_id	NHOST	NSCOPE	Hostname
3ffe:501::1	0	0	-	"foo.kame.net"
3ffe:501::1	0	1	-	"3ffe:501::1"
fec0::1	10	-	-	"fec0::1%10"
fe80::1	5	-	0	"fe80::1%ne0"
fe80::1	5	-	1	"fe80::1%5"

Non-typical cases for getaddrinfo():

Hostname	NI_NUMERICHOST	sin6_addr	sin6_scope_id	N/B
"foo.kame.net"	1	Error		
"foo.kame.net%20"	-	Error		(*1)

"foo.kame.net%none"	-	Error		(*1)
"3ffe:501::1%none"	-	Error		
"3ffe:501::1%0"	-	3ffe:501::1	0	(*2)
"3ffe:501::1%20"	-	3ffe:501::1	20	(*2)
"fec0::1%none"	-	Error		
"fec0::1"	-	fec0::1	0	(*3)
"fec0::1%0"	-	fec0::1	0	(*4)
"fec0::1%20"	-	fec0::1	20	(*5)
"fe80::1%none"	-	Error		
"fe80::1"	-	fe80::1	0	(*3)
"fe80::1%0"	-	fe80::1	0	(*4)
"fe80::1%20"	-	fe80::1	20	(*5)

(*1) <zone_id> against an FQDN is invalid.
(*2) We do not expect that <zone_id> is specified for a global address, but we don't regard it as invalid.
(*3) We usually expect that a scoped address is specified with <zone_id>, but if no identifier is specified we just set 0 to the sin6_scope_id field.
(*4) Explicitly specifying 0 as <zone_id> is not meaningful, but we just treat the value as opaque.
(*5) The <zone_id> portion is opaque to getaddrinfo() even if it is invalid. It is kernel's responsibility to raise errors, if there is any connection attempt that the kernel cannot handle.

Non-typical cases for getnameinfo():

sin6_addr	sin6_scope_id	NHOST	NSCOPE	Hostname	N/B
3ffe:501::1	20	1	-	"3ffe:501::1%20"	(*6)
3ffe:501::1	20	0	-	"foo.kame.net"	(*7)
fec0::1	20	-	-	"fec0::1%20"	
fec0::1	0	-	-	"fec0::1"	(*8)
fe80::1	20	-	-	"fe80::1%20"	
fe80::1	0	-	-	"fe80::1"	(*8)

(*6) We do not expect that a global IPv6 address has a non-zero zone identifier. But if it is the case, we just treat it as opaque.
(*7) Despite the above, if the NI_NUMERICHOST is clear, we resolve the address to a hostname and print the name without scope zone information. We might have to reconsider this behavior.
(*8) We usually expect that a scoped address has a non-zero zone identifier. But if the identifier is 0, we simply print the address portion without scope zone information.

Acknowledgments

We authors are indebted to Brian Zill, Richard Draves, and Francis Dupont for their careful comments and suggestions in a discussion to define a unified format among early implementations.

Jim Bound also gave us valuable comments and clarifications through discussions about API extensions for scoped addresses in the ipngwg mailing list.

Hideaki YOSHIFUJI commented on relationship between the URI syntax and the proposed format, and helped us improve the related section.

Jun-ichiro Hagino has been helping us through all the discussions and his implementation efforts.

Authors' Addresses

Tatuya JINMEI
Research and Development Center, Toshiba Corporation
1 Komukai Toshiba-cho, Kawasaki-shi
Kanagawa 212-8582, JAPAN
Tel: +81-44-549-2230
Fax: +81-44-520-1841
Email: jinmei@isl.rdc.toshiba.co.jp

Atsushi Onoe
Internet Systems Laboratory, IN Laboratories, Sony Corporation
6-7-35 Kitashinagawa, Shinagawa-ku, Tokyo 141-0001, JAPAN
Tel: +81-3-5448-4620
Fax: +81-3-5448-4622
Email: onoe@sm.sony.co.jp

References

[ADDRARCH] Hinden, R., Deering, S., "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

[BASICAPI] Gilligan, R. E., Thomson, S., Bound, J., Stevens, W., "Basic Socket Interface Extensions for IPv6", Internet-Draft, May 2000, <draft-ietf-ipngwg-rfc2553bis-00.txt>.

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

[SCOPEARCH] S. Deering, B. Haberman, B. Zill, "IP Version 6 Scoped Address Architecture", Internet-Draft, March 2000, <draft-ietf-ipngwg-scoping-arch-00.txt>.

[SCOPEDROUTING] Haberman, B., "Routing of Scoped Addresses in the Internet Protocol Version 6 (IPv6)", Internet-Draft, March 2000, <draft-ietf-ipngwg-scoped-routing-03.txt>.

[STD-PROC] Bradner, S., The Internet Standards Process -- Revision 3, RFC 2026, October 1996.

[URLFORMAT] Hinden, R., Carpenter, B., Masinter, L., "Preferred Format for Literal IPv6 Addresses in URL's", RFC 2732, December 1999.

[URI] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.