

Advanced Sockets API for IPv6

<draft-ietf-ipngwg-rfc2292bis-03.txt>

**W. Richard Stevens, Matt Thomas,
Erik Nordmark, Tatuya Jinmei**

Changes From 02

- new stuff
 - (new) MTU APIs
 - traffic class API
 - merged from a separate draft

- removed definitions on ongoing specs
 - site prefixes, mobile IPv6

- API clarifications
 - receipt of optional info on a TCP socket
 - extension header manipulation
 - sticky option overriding rule
 - option ordering (out IF selection, hoplimit)
 - ...

Receipt of TCP optional info (1/3)

□ A brief history

- RFC 2292: use `getsockopt()`
 - get the whole info at a certain point of time by a single `getsockopt()`
- 2292bis-00, 01, 02: `recvmsg()` + ancillary data
 - for data segments
 - apps can catch changes on the optional info

□ Problems on the new spec

- "send-only" applications cannot receive optional info
- it cannot catch all the changes
 - the info cannot be used for security purposes

□ Reconsideration of the API spec

- it is not feasible to try to catch all the changes
- an application will get the info only once (or a few times)
- all TCP apps should be able to get the info
 - including send-only ones

Receipt of TCP optional info (2/3)

□ Alternative 1: recvmsg() + ancillary data only

- allow recvmsg() to return 0 with ancillary data objects

- pro: easy migration from 2292bis-02 applications

- pro: less impact on existing 2292bis-02 kernels

- con: hard migration from RFC2292 applications

- con: impact on existing RFC2292 kernels

- con: impact on the traditional socket semantics

 - e.g. need additional condition to detect an end of connection

```
if ((n = recvmsg(s, &msghdr, 0)) == 0
```

```
    && msghdr.msg_controllen == 0)
```

```
    close(s);
```

 - e.g. what if the app calls shutdown(0) (i.e. unreadable)?

Receipt of TCP optional info (3/3)

□ Alternative 2 (in 03): RFC2292-style socket

option

- pro: easy migration from RFC 2292 applications
- pro: less impact on existing RFC 2292 kernels
- pro: less impact on the traditional socket semantics

- con: hard migration from rfc2292bis-02 applications
- con: impact on existing rfc2292bis-02 kernels

□ Alternative 1 vs 2

- apart from the socket semantics, it's a migration
(compatibility) issue.
- need to reach a consensus based on implementors' sense

Extension Header Manipulation

- Based on a discussion in the wg ML (June 2001)
 - provide minimum flexibility to support header ordering
 - for full flexibility, use other APIs

- Header types
 - HbH, Dest, and Routing (like in RFC2292)
 - do not handle IPsec headers, Fragment headers, etc

- Header ordering
 - Outgoing: assume the RFC2460 ordering
 - Incoming: no assumption
 - MIP6 packet may break the recommended ordering
 - all headers will appear as ancillary objects in the received order
 - IPV6_RECVRTHDRDSTOPTS is obsoleted as an effect

Sticky Option Overriding

- Before 03 (incl. RFC 2292)
 - an ancillary data item disables all sticky options

- In 03
 - an ancillary data item only disables a same type of sticky option
 - e.g. ancillary IPV6_RTHDR disables sticky IPV6_RTHDR but not sticky IPV6_HOPOPTS

- The rationale of the change
 - each sticky option can be specified as a single socket option
 - there is an ancillary only option (IPV6_REACHCONF)
 - there is an application that uses an ancillary data item as well as a sticky option

IPV6_HOPLIMIT Sticky Option

- 3 (sticky) options to specify the hop limit
 - IPV6_UNICAST_HOPS, IPV6_MULTICAST_HOPS, IPV6_HOPLIMIT
 - the ordering was unclear

- 03 disables the usage of sticky IPV6_HOPLIMIT
 - pros: no ambiguity about the ordering, simple usage
 - cons: backward compatibility

Specifying The Outgoing Interface

- 3 options that affect the out IF
 - IPV6_PKTINFO, IPV6_NEXTHOP, IPV6_MULTICAST_IF
 - the ordering among the options was unclear

- 03 clarifies the ordering of the options
 - 1. IPV6_PKTINFO
 - 2. IPV6_MULTICAST_IF (for multicast destinations)
 - 3. IPV6_NEXTHOP (the IF to the next hop)

- The point is the deterministic behavior, not the ordering itself

Path MTU API

- `IPV6_PATHMTU`
 - before 03: data is an integer (MTU value)
 - in 03: data is a structure to support non-connected sockets

- `IPV6_DONTFRAG` socket option to skip fragmentation

- `IPV6_PATHMTU` socket option to tell apps the current path MTU
 - for connected socket only
 - `getsockopt()` does not take an argument value
 - typical cases should be covered

rcmd API

History

- a discussion in 1996
 - a proposal from Carl Williams@Sun
 - basic API vs advanced API
 - Craig Metz suggested the adv. API
- 2292bis-00 introduced a new section for rcmd (1999)
 - rresvport_af(), rcmd_af(), rexec_af()
- no update for two years (since 2292bis-01)

What should we do this?

- just describe the motivation and complete it?

Next Step

- Revise the draft
 - Collect comments
 - Fix open issues
 - TCP implications, rcmd issues, others (if any)
 - No new stuff will be included
 - fix the API ASAP, provide uniform API