Developer Summit
Meeting of the Network Stack Cabal

ip6_{in,out,..}

Bjoern A. Zeeb
bz@FreeBSD.org

The FreeBSD Project

BSDCAN - The Technical BSD Conference, 2007

Overview

- Most know ip_*, but ip6_* often ignored
- IPv6 got press because of debatable design specs lately
- But code also needs cleanup, more hands

IPv4 vs. IPv6 LOC

- Long(er) functions (screen length):

func	line# v4	line# v6	difference
input	218- 666 (448)	219- 772 (553)	+105 (+23%)
forward	1246-1436 (190)	100- 659 (559)	+369 (+194%)
output	109- 616 (507)	152-1172 (1020)	+513 (+101%)

- Why is that?
  - IPsec not factored out
  - More in-function multicast handling
  - Scope handling
  - Fragmentation handling and Extension Headers (output path)

IPv6 Header(s)

```
struct ip6_hdr {
  union {
  struct ip6_hdrctl {
    u_int32_t ip6_un1_flow; /*20bits*/
    u_int16_t ip6_un1_plen;
    u_int8_t  ip6_un1_nxt;
    u_int8_t  ip6_un1_hlim;
  } ip6_un1;
  u_int8_t ip6_un2_vfc;
  }
  struct in6_addr ip6_src;
  struct in6_addr ip6_dst;
}
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |    Flow Label                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header |  Hop Limit    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
:                     Source Address                            :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
:                  Destination Address                          :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- No variable length or optional fields, no checksum, no fragment offset, . . .
- But ‘Next Header’ for ‘flexibility’:
  - Hop-by-Hop Options Header (`struct ip6_hbh`)
  - Routing Header (`struct ip6_rthdr`)
  - Fragment Header (`struct ip6_frag`)
  - Destination Options Header (`struct ip6_hbh`)
  - AH, ESP, UDP, TCP, . . .

Code to inspect

- `netinet/ip6.h`
  IPv6 and Extension Header definitions
- `netinet6/ip6_input.c`
  ip6_input(), ip6_hopopts_input(), ip6_process_hopopts()
- `netinet6/ip6_forward.c`
  ip6_forward()
- `netinet6/ip6_output.c`
  ip6_output(), ip6_copyexthdr(), ip6_splithdr(), ip6_insert_jumboopt(), ip6_insertfraghdr()

Going to concentrate on the special parts of ip6_input() and ip6_output() in contrast to ip_* counterparts [marked with a (*) in slides following].

ip6_input

1. GIANT_REQUIRED /* XXX for now */
2. Cleanup mbuf flags
3. Update mbuf statistics
4. No IPv6 processing if no IPv6 configured (*)
5. Interface stats
6. mbuf gymnastics, that need review (*)
7. IP6_EXTHDR_CHECK macro, more mbuf gymnastics (*)
8. Check for IPv6 version
9. Check src/dst addresses to make sense (*)
10. ALTQ processing
11. Check for v4 mapped addresses (*)
12. PFIL processing
13. Scope validation (*)

ip6_input

cont.ed

14 Multicast check (or fall through)

15 Unicast check (or fall through)

16 Special forwarding checks (or fall through)

17 FAITH check (or fall through) (*)

18 Not for us, are we forwarding, else drop

19 Save 'my dst address'

20 Process Hop-by-Hop Option Headers if present (*)

21 Validate mbuf data length left

22 Forward (multicast or unicast) if not for us

23 Redo 17 for unknown reasons? (*)

24 Loop through next headers until everything was handled (*)

ip6_output

1. Prepare extension headers if to be present (*)
2. IPsec policy selection
3. Extension header and payload separation (*)
4. Lengths and jumbogram Hop-by-Hop option (*)
5. Assemble packet, set next header (*)
6. IPsec transport mode processing
7. Routing Header 0 - soon to be obsolete code(?) (*)
8. Source address validation
9. Get destination

ip6_output

cont.ed

10 Set traffic class flags
11 Update Hop-Limit
12 IPsec tunnel mode processing
13 Select route and outgoing interface
14 Multicast checks
15 MTU handling
16 Clear (internal) scope (*)
17 Process (our own) Hop-By-Hop Extension Header if present (*)
18 PFIL processing
19 Check if destination changed: yes? to ourselves (ip_input is next, else start over at 10)
20 Fragmentation if needed (*)
21 Update statistics and send via nd6_output() (*)
22 Cleanup

TODO

- Cleanup:
  - Get fast_ipsec/IPv6 in and remove KAME IPsec stack
  - Cleanup obvious parts, like duplicate code
  - Factor out more code where possible?
  - Better ‘assimilation’ to reduce difference to IPv4 code
  - Review mbuf constraints - are they still true?
  - . . .
- Make sure we will be able to pass out v4 packets from v6 stack
- More hands, IPv6 is there, do not ignore it