Chapitre 20

Implémentation Linux de IPv4

Nous allons commencer à étudier le début de l'implémentation de la couche IPv4 dans ce chapitre, à savoir l'en-tête IPv4 le calcul de la somme de contrôle.

20.1 Implémentation Linux de l'en-tête IP

L'en-tête IP est une entité du type `struct iphdr`. Celui-ci est défini dans le fichier en-tête `linux/include/linux/ip.h`:

Code Linux 2.6.10

```
6   *              Definitions pour le protocole IP.
7   *
8   * Version :    @(#)ip.h        1.0.2   04/28/93
9   *
10  * Auteurs :    Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
[...]
16  */
17 #ifndef _LINUX_IP_H
18 #define _LINUX_IP_H
19 #include <asm/byteorder.h>
[...]
167 struct iphdr {
168 #if defined(__LITTLE_ENDIAN_BITFIELD)
169         __u8    ihl:4,
170                 version:4;
171 #elif defined (__BIG_ENDIAN_BITFIELD)
172         __u8    version:4,
173                 ihl:4;
174 #else
175 #error  "Please fix <asm/byteorder.h>"
176 #endif
177         __u8    tos;
```

```
178         __u16   tot_len;
179         __u16   id;
180         __u16   frag_off;
181         __u8    ttl;
182         __u8    protocol;
183         __u16   check;
184         __u32   saddr;
185         __u32   daddr;
186         /* Les options debutent ici. */
187 };
```

Les drapeaux de cet en-tête IP sont définis comme constantes symboliques dans le en-tête `linux/include/net/ip.h`:

Code Linux 2.6.10

```
69 /* Drapeaux IP. */
70 #define IP_CE           0x8000          /* Drapeau : "Congestion"           */
71 #define IP_DF           0x4000          /* Drapeau : "Don't Fragment"       */
72 #define IP_MF           0x2000          /* Drapeau : "More Fragments"       */
73 #define IP_OFFSET       0x1FFF          /* partie "Decalage de fragment"    */
```

Les types de service sont définis comme constantes symboliques dans le fichier en-tête `linux/include/linux/ip.h`:

Code Linux 2.6.10

```
21 #define IPTOS_TOS_MASK                  0x1E
22 #define IPTOS_TOS(tos)                  ((tos)&IPTOS_TOS_MASK)
23 #define IPTOS_LOWDELAY                  0x10
24 #define IPTOS_THROUGHPUT                0x08
25 #define IPTOS_RELIABILITY               0x04
26 #define IPTOS_MINCOST                   0x02
27
28 #define IPTOS_PREC_MASK                 0xE0
29 #define IPTOS_PREC(tos)                 ((tos)&IPTOS_PREC_MASK)
30 #define IPTOS_PREC_NETCONTROL           0xe0
31 #define IPTOS_PREC_INTERNETCONTROL      0xc0
32 #define IPTOS_PREC_CRITIC_ECP           0xa0
33 #define IPTOS_PREC_FLASHOVERRIDE        0x80
34 #define IPTOS_PREC_FLASH                0x60
35 #define IPTOS_PREC_IMMEDIATE            0x40
36 #define IPTOS_PREC_PRIORITY             0x20
37 #define IPTOS_PREC_ROUTINE              0x00
```

20.2 Calcul de la somme de contrôle IP

Le calcul de la somme de contrôle IP est effectué par la fonction `ip_fast_csum()`. L'implémentation de celle-ci dépend de l'architecture. Pour les microprocesseurs Intel, elle est définie dans le fichier `linux/include/asm-i386/checksum.h`:

Code Linux 2.6.10

```
54 /*
55  *      Ceci est une version de ip_compute_csum() optimisee pour les en-tetes IP,
56  *      qui calcule toujours sur des frontieres de 4 octets.
57  *
58  *      Par Jorge Cwik <jorge@laser.satlink.net>, adapte pour linux par
59  *      Arnt Gulbrandsen.
60  */
61 static inline unsigned short ip_fast_csum(unsigned char * iph,
62                                           unsigned int ihl)
63 {
64         unsigned int sum;
65
66         __asm__ __volatile__(
67             "movl (%1), %0      ;\n"
```

```
68              "subl $4, %2        ;\n"
69              "jbe 2f             ;\n"
70              "addl 4(%1), %0     ;\n"
71              "adcl 8(%1), %0     ;\n"
72              "adcl 12(%1), %0    ;\n"
73 "1:          adcl 16(%1), %0     ;\n"
74              "lea 4(%1), %1      ;\n"
75              "decl %2            ;\n"
76              "jne 1b             ;\n"
77              "adcl $0, %0        ;\n"
78              "movl %0, %2        ;\n"
79              "shrl $16, %0       ;\n"
80              "addw %w2, %w0      ;\n"
81              "adcl $0, %0        ;\n"
82              "notl %0            ;\n"
83 "2:                             ;\n"
84         /* Puisque les registres d'entree qui sont charges avec iph et ipl
85            sont modifies, nous devons aussi les specifier comme sortie, sinon gcc
86            supposera qu'ils contiennent les valeurs originelles. */
87         : "=r" (sum), "=r" (iph), "=r" (ihl)
88         : "1" (iph), "2" (ihl)
89         : "memory");
90         return(sum);
91 }
```