Programowanie sieciowe
Nieprzetworzone gniazda sieciowe

Michał Kalewski

Instytut Informatyki
Politechnika Poznańska
ul. Piotrowo 2, 60–965 Poznań
mkalewski@cs.put.poznan.pl
http://www.cs.put.poznan.pl/mkalewski

Poznań · 16 kwietnia 2019 r.

Plan wykładu

1 Nieprzetworzone gniazda sieciowe

2 Pakiety transmitowane i pakiety odbierane

3 Żądania funkcji systemowej `ioctl(2)` dla gniazd sieciowych

4 Podsumowanie

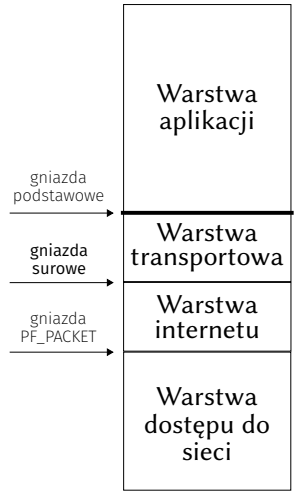
5 Pytania i zadania

Nieprzetworzone gniazda sieciowe

Nieprzetworzone gniazda sieciowe (1/2)

Nieprzetworzone (lub inaczej *surowe*) gniazda sieciowe pozwalają na dostęp programistyczny do warstwy sieciowej modelu warstwowego (internetowej w warstwowym modelu internetowym).

Z wykorzystaniem nieprzetworzonych gniazd sieciowych możliwe jest transmitowanie i odbieranie pakietów własnych protokołów *routowanych* (lub inaczej *trasowanych*), tj. protokołów realizujących komunikację pomiędzy komputerami, które nie są ze sobą bezpośrednio fizycznie połączone.



Rysunek 1: Surowe gniazda sieciowe w odniesieniu do warstwowego modelu internetowego.

Nieprzetworzone gniazda sieciowe (2/2)

Nieprzetworzone gniazda sieciowe dostarczają programistom trzy podstawowe funkcjonalności:

- możliwość transmitowania i odbierania pakietów protokołów IGMP, ICMP oraz ICMPv6 — aplikacje wykorzystujące te protokoły mogą być implementowane poza jądrem systemu operacyjnego;
- możliwość transmitowania pakietów protokołów IPv4 oraz IPv6 z wartościami pól `Protocol`/`NextHeader`, które nie są przetwarzane przez jądro systemu operacyjnego (jądra systemów operacyjnych UNIX oraz GNU/Linux przetwarzają najczęściej jedynie pakiety z wartościami 1/ICMP, 2/IGMP, 6/TCP oraz 17/UDP);
- możliwość transmitowania pakietów z własnymi wartościami pól nagłówka.

Tworzenie nieprzetworzonych gniazd sieciowych (1/5)

```
int sfd = socket(PF_INET, SOCK_RAW, int protocol);
```

Wartości trzeciego parametru wywołania funkcji systemowej `socket(2)` określają stałe `IPPROTO_*` zdefiniowane w pliku nagłówkowym `netinet/in.h`. *zob.* `ip(7)`

PRZYKŁAD:

```
$ cat netinet/in.h | grep -v "^#" | grep "IPPROTO_"
IPPROTO_IP   = 0,   /* Dummy protocol for TCP.             */
IPPROTO_ICMP = 1,   /* Internet Control Message Protocol.  */
IPPROTO_IGMP = 2,   /* Internet Group Management Protocol. */
IPPROTO_IPIP = 4,   /* IPIP tunnels.                       */
IPPROTO_TCP  = 6,   /* Transmission Control Protocol.      */
IPPROTO_EGP  = 8,   /* Exterior Gateway Protocol.          */
IPPROTO_PUP  = 12,  /* PUP protocol.                       */
IPPROTO_UDP  = 17,  /* User Datagram Protocol.             */
IPPROTO_IDP  = 22,  /* XNS IDP protocol.                   */
```

https://github.com/torvalds/linux/blob/master/include/uapi/linux/in.h

Tworzenie nieprzetworzonych gniazd sieciowych (2/5)

Nagłówki pakietów IPv4, transmitowanych nieprzetworzonymi gniazdami sieciowymi, tworzone są automatycznie.

Tworzenie własnych nagłówków dla pakietów transmitowanych nieprzetworzonymi gniazdami sieciowymi możliwe jest po aktywowaniu opcji `IP_HDRINCL`.

Pakiety odbierane z nieprzetworzonych gniazd sieciowych zawsze zawierają nagłówki.

Tworzenie nieprzetworzonych gniazd sieciowych (3/5)

Dla nieprzetworzonych gniazd sieciowych możliwe jest aktywowanie opcji `IP_HDRINCL` przy użyciu funkcji systemowej `setsockopt(2)`:

```
1 const int on = 1;
2 setsockopt(sfd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on));
```

Alternatywnie, w systemach operacyjnych GNU/Linux ten sam efekt można osiągnąć podając jako trzeci argument wywołania funkcji systemowej `socket(2)` stałą `IPPROTO_RAW`.

PRZYKŁAD:

```
$ cat netinet/in.h | grep -v "^#" | grep "IPPROTO_RAW"
IPPROTO_RAW  = 255,  /* Raw IP packets.                    */
```

zob. `raw(7)`

Tworzenie nieprzetworzonych gniazd sieciowych (4/5)

Wywołanie funkcji systemowej `bind(2)` dla deskryptora nieprzetworzonego gniazda sieciowego jest możliwe, lecz jest to rozwiązanie rzadko stosowane.

Dla nieprzetworzonych gniazd sieciowych bez aktywowanej opcji `IP_HDRINCL`, wywołanie funkcji systemowej `bind(2)` umożliwia ustalenie adresu IP nadawcy.

Jeżeli funkcja systemowa `bind(2)` nie zostanie wywołana, jądro systemu operacyjnego nada wysyłanemu pakietowi adres IP nadawcy równy adresowi skonfigurowanemu na interfejsie sieciowym, przez który pakiet ten będzie transmitowany.

Tworzenie nieprzetworzonych gniazd sieciowych (5/5)

Strukturą adresową wykorzystywaną przez nieprzetworzone gniazda sieciowe jest struktura `sockaddr_in`.

Na warstwie sieciowej nie wykorzystuje się numerów portów, więc pole `sin_port` powinno zawierać wartość 0 (dla pakietów odbieranych, wartość pola `sin_port` zawsze ma wartość 0).

zob. `ip(7)`

Wywołanie funkcji systemowej `connect(2)` dla deskryptora nieprzetworzonego gniazda sieciowego jest możliwe, lecz jest to rozwiązanie rzadko stosowane.

Jedyną korzyścią z wywołania funkcji systemowej `connect(2)` jest ustalenie adresu IP odbiorcy pakietów, co pozwala na wykorzystywanie np. funkcji systemowej `write(2)` w miejsce `sendto(2)`.

Struktura nagłówka pakietów IPv4

Struktura nagłówka pakietu IPv4 zdefiniowana jest w pliku nagłówkowym `netinet/ip.h`.

```
1  struct iphdr {
2    __u8    ihl:4,      /* Internet Header Length, 4 bits */
3            version:4;  /* version, 4 bits */
4    __u8    tos;        /* type of service */
5    __be16  tot_len;    /* total length */
6    __be16  id;         /* identification */
7    __be16  frag_off;   /* fragment offset */
8    __u8    ttl;        /* Time To Live */
9    __u8    protocol;   /* protocol */
10   __sum16 check;      /* header checksum */
11   __be32  saddr;      /* source address */
12   __be32  daddr;      /* destination address */
13 };
```

https://github.com/torvalds/linux/blob/master/include/uapi/linux/ip.h

Suma kontrolna nagłówka pakietu IPv4 (1/2)

Przykład nr 1 — tworzenie:

Nagłówek pakietu IP:
`4500 0073 0000 4000 4011 XXXX c0a8 0001 c0a8 00c7`

Suma wartości:
`4500 + 0073 + 0000 + 4000 + 4011 + c0a8 + 0001 + c0a8 + 00c7 = 2479c`

Konwersja do liczby binarnej: `2479c = 0010 0100 0111 1001 1100`

Suma pierwszych czterech bitów z pozostałymi:
`0010 + 0100 0111 1001 1100 = 0100 0111 1001 1110`

Negacja binarna: `1011 1000 0110 0001`

Wynik: `1011 1000 0110 0001 = b861`

Nagłówek pakietu IP:
`4500 0073 0000 4000 4011` **`b861`** `c0a8 0001 c0a8 00c7`

Suma kontrolna nagłówka pakietu IPv4 (2/2)

Przykład nr 2 — weryfikacja:

Nagłówek pakietu IP:
`4500 0073 0000 4000 4011 b861 c0a8 0001 c0a8 00c7`

Suma wartości:
`4500 + 0073 + 0000 + 4000 + 4011 + b861 + c0a8 + 0001 + c0a8 + 00c7 = 2fffd`

Konwersja do liczby binarnej: `2fffd = 0010 1111 1111 1111 1101`

Suma pierwszych czterech bitów z pozostałymi:
`0010 + 1111 1111 1111 1101 = 1111 1111 1111 1111`

Negacja binarna: `0000 0000 0000 0000`

Pakiety transmitowane i pakiety odbierane

Pakiety transmitowane

Pakiety transmitowane nieprzetworzonymi gniazdami sieciowymi przetwarzane są w sposób następujący:

- jeżeli opcja `IP_HDRINCL` nie została aktywowana, to jądro systemu operacyjnego utworzy nagłówek protokołu IPv4 oraz ustali wartość pola `Protocol` zgodnie z trzecim argumentem wywołania funkcji systemowej `socket(2)`
  (w tym przypadku, dane przekazywane przez programistę nie zawierają nagłówka);
- jeżeli opcja `IP_HDRINCL` została aktywowana, to programista zobowiązany jest utworzyć samodzielnie nagłówek protokołu IP
  (w tym przypadku, dane przekazywane przez programistę zawierają nagłówek);
- jądro systemu operacyjnego samodzielnie dokonuje fragmentacji pakietów, których rozmiar jest większy niż MTU interfejsu sieciowego wykorzystywanego do ich transmisji.

Pakiety odbierane (1/3)

Pakiety odbierane z interfejsów sieciowych przekazywane są do nieprzetworzonych gniazd sieciowych zgodnie z następującymi zasadami ogólnymi:

- odbierane datagramy UDP oraz segmenty TCP nigdy nie są przekazywane;
- wszystkie pakiety protokołu ICMP są przekazywane po tym, jak jądro systemu operacyjnego zakończy ich przetwarzanie (wyjątek stanowić mogą pakiety *echo request*, *timestamp request* oraz *address mask request*);
- wszystkie pakiety protokołu IGMP są przekazywane po tym, jak jądro systemu operacyjnego zakończy ich przetwarzanie;
- wszystkie pakiety protokołu IP są przekazywane, jeżeli wartość pola `Protocol` wskazuje na protokół, który nie jest obsługiwany przez jądro systemu operacyjnego.

Pakiety odbierane (2/3)

Przed dostarczeniem pakietu do nieprzetworzonego gniazda sieciowego wykonywane są następujące testy szczegółowe:

- porównywana jest wartość trzeciego argumentu wywołania funkcji systemowej `socket(2)` z wartością pola `Protocol` odebranego pakietu;
- porównywany jest adres IP odbiorcy w odebranym pakiecie z adresem lokalnym powiązanym z gniazdem funkcją systemową `bind(2)`;
- porównywany jest adres nadawcy odebranego pakietu z adresem zdalnym powiązanym z gniazdem funkcją systemową `connect(2)`.

Odebrany pakiet przekazywany jest do wszystkich nieprzetworzonych gniazd sieciowych, dla których wszystkie trzy powyższe testy wykonane zostały poprawnie.

Pakiety odbierane (3/3)

Nieprzetworzone gniazdo sieciowe utworzone przy pomocy funkcji systemowej `socket(2)` z trzecim argumentem wywołania równym 0, dla którego nie wywołano funkcji systemowych `bind(2)` oraz `connect(2)`, odbierać będzie wszystkie pakiety jakie jądro systemu operacyjnego przekazuje do nieprzetworzonych gniazd sieciowych.

Pakiety protokołu IPv4 odbierane z nieprzetworzonych gniazd sieciowych zawsze zawierają nagłówek.

</> Transmisja pakietu IPv4

```
#define IPPROTO_CUSTOM 222
```

```
int sfd;
struct sockaddr_in addr;

sfd = socket(PF_INET, SOCK_RAW, IPPROTO_CUSTOM);
memset(&addr, 0, sizeof(addr));
addr.sin_family = AF_INET;
addr.sin_port = 0;
addr.sin_addr.s_addr = inet_addr(argv[1]);
sendto(sfd, argv[2], strlen(argv[2]) + 1, 0,
       (struct sockaddr*) &addr, sizeof(addr));
close(sfd);
```


</> Odbiór pakietu IPv4

```
int sfd, rc;
char buf[65536], saddr[16], daddr[16];
char *data;
socklen_t sl;
struct sockaddr_in addr;
struct iphdr *ip;

sfd = socket(PF_INET, SOCK_RAW, IPPROTO_CUSTOM);
memset(&addr, 0, sizeof(addr));
sl = sizeof(addr);
rc = recvfrom(sfd, buf, sizeof(buf), 0,
              (struct sockaddr*) &addr, &sl);
ip = (struct iphdr*) &buf;
inet_ntop(AF_INET, &ip->saddr, (char*) &saddr, 16);
inet_ntop(AF_INET, &ip->daddr, (char*) &daddr, 16);
data = (char*) ip + (ip->ihl * 4);
close(sfd);
```

Żądania funkcji systemowej `ioctl(2)` dla gniazd sieciowych

Żądania funkcji systemowej `ioctl(2)` dla gniazd sieciowych

`SIOCGSTAMP` Zwraca, w strukturze `timeval`, czas, w którym przekazano użytkownikowi ostatni pakiet z gniazda.

`SIOCSPGRP` Ustala identyfikator procesu (lub grupy procesów), do których wysyłane są sygnały gniazda (`SIGIO` oraz `SIGURG`).

`SIOCGPGRP` Zwraca identyfikator procesu (lub grupy procesów), do których wysyłane są sygnały gniazda.

`FIOASYNC` Ustala synchroniczny lub asynchroniczny tryb działania gniazda.

Podsumowanie

Podsumowanie

- Tworzenie nieprzetworzonych gniazd sieciowych;
  - stałe `IPPROTO_*`;
  - opcja `IP_HDRINCL`;
  - struktura `iphdr`;
  - suma kontrolna nagłówka pakietu IPv4.
- Obsługa pakietów transmitowanych oraz pakietów odbieranych.
- Żądania funkcji systemowej `ioctl(2)` do obsługi gniazd sieciowych.

Pytania i zadania

Pytania i zadania — praca własna

1. Podobną funkcjonalność do nieprzetworzonych gniazd sieciowych można uzyskać z zastosowaniem gniazd sieciowych `PF_PACKET`, jaką zatem korzyść dają programiście gniazda surowe?
2. Transmisję własnych pakietów warstwy trzeciej można zrealizować także za pomocą biblioteki `libnet`, jakie są zatem korzyści z użycia gniazd surowych zamiast tej biblioteki?
3. Zapoznaj się z pełną listą protokołów transportowanych pakietami IP, które są rozpoznawane przez systemy operacyjne UNIX oraz GNU/Linux (*zob.* `netinet/in.h`).
4. Sprawdź na czym polega polega mechanizm skanowania typu *idle scan*.

Dziękuję za uwagę!