Programowanie sieciowe
Internet Protocol version 6

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 · 14 maja 2019 r.

Plan wykładu

Gniazda sieciowe protokołu IPv6

Gniazda sieciowe protokołu IPv6

- Gniazda strumieniowe TCP:

```
int socket(PF_INET6, SOCK_STREAM, 0);
```

- Gniazda datagramowe UDP:

```
int socket(PF_INET6, SOCK_DGRAM, 0);
```

- Gniazda protokołu SCTP:

```
int socket(PF_INET6, SOCK_STREAM, IPPROTO_SCTP);
```

- Gniazda nieprzetworzone:

```
int socket(PF_INET6, SOCK_RAW, int protocol);
```

zob. `ipv6(7)`

Struktura adresowa sockaddr_in6

```c
struct sockaddr_in6 {
  sa_family_t     sin6_family;   /* AF_INET6 */
  in_port_t       sin6_port;     /* port number */
  uint32_t        sin6_flowinfo; /* IPv6 flow information */
  struct in6_addr sin6_addr;     /* IPv6 address */
  uint32_t        sin6_scope_id; /* Scope ID (new in 2.4) */
};

struct in6_addr {
  unsigned char   s6_addr[16];   /* IPv6 address */
};
```

zob. ipv6(7) oraz in6.h

https://github.com/torvalds/linux/blob/master/include/uapi/linux/in6.h

Implementacje zależne od protokołu warstwy sieciowej

</> Klient TCP — IPv4

```c
int sfd;
char buf[128];
struct sockaddr_in saddr;

memset(&saddr, 0, sizeof(saddr));
saddr.sin_family = AF_INET;
saddr.sin_port = htons(atoi(argv[2]));
inet_pton(AF_INET, argv[1], &saddr.sin_addr);
sfd = socket(PF_INET, SOCK_STREAM, 0);
connect(sfd, (struct sockaddr*) &saddr, sizeof(saddr));
read(sfd, buf, sizeof(buf));
close(sfd);
```


</> Klient TCP — IPv6

```c
int sfd;
char buf[128];
struct sockaddr_in6 saddr;

memset(&saddr, 0, sizeof(saddr));
saddr.sin6_family = AF_INET6;
saddr.sin6_port = htons(atoi(argv[2]));
inet_pton(AF_INET6, argv[1], &saddr.sin6_addr);
sfd = socket(PF_INET6, SOCK_STREAM, 0);
connect(sfd, (struct sockaddr*) &saddr, sizeof(saddr));
read(sfd, buf, sizeof(buf));
close(sfd);
```

</> Serwer TCP — IPv4

```c
int sfd, cfd;
socklen_t sl;
struct sockaddr_in saddr, caddr;
memset(&saddr, 0, sizeof(saddr));
saddr.sin_family = AF_INET;
saddr.sin_addr.s_addr = INADDR_ANY;
saddr.sin_port = htons(1234);
sfd = socket(PF_INET, SOCK_STREAM, 0);
bind(sfd, (struct sockaddr*) &saddr, sizeof(saddr));
listen(sfd, 5);
while(1) {
  sl = sizeof(caddr);
  cfd = accept(sfd, (struct sockaddr*) &caddr, &sl);
  write(cfd, "Hello World!\n", 13);
  close(cfd);
}
```


</> Serwer TCP — IPv4/IPv6

```c
int sfd, cfd;
socklen_t sl;
struct sockaddr_in6 saddr, caddr;
memset(&saddr, 0, sizeof(saddr));
saddr.sin6_family = AF_INET6;
saddr.sin6_addr = in6addr_any;  // in6addr_loopback
saddr.sin6_port = htons(1234);
sfd = socket(PF_INET6, SOCK_STREAM, 0);
bind(sfd, (struct sockaddr*) &saddr, sizeof(saddr));
listen(sfd, 5);
while(1) {
  sl = sizeof(caddr);
  cfd = accept(sfd, (struct sockaddr*) &caddr, &sl);
  write(cfd, "Hello World!\n", 13);
  close(cfd);
}
```


Serwer TCP — IPv4/IPv6 — uwagi

Fragment dokumentacji `ipv6(7)`:

IPv4 connections can be handled with the v6 API by using the v4-mapped-on-v6 address type; thus a program needs to support only this API type to support both protocols.

- Mapowanie adresów *v4-mapped-on-v6*:
  80 bitów `0`, 16 bitów `1` i adres IPv4,
  np.: `::ffff:192.168.1.1`.
- Akceptacja połączeń jedynie od adresów IPv6 wymaga ustawienia opcji `IPV6_V6ONLY` dla głównego gniazda sieciowego serwera funkcją systemową `setsockopt(2)`.

Implementacje niezależne od protokołu warstwy sieciowej

Funkcja `getaddrinfo(3)`

```
int getaddrinfo(const char *node, const char *service,
                const struct addrinfo *hints,
                struct addrinfo **res);
```

```
1  struct addrinfo {
2    int              ai_flags;
3    int              ai_family;
4    int              ai_socktype;
5    int              ai_protocol;
6    socklen_t        ai_addrlen;
7    struct sockaddr *ai_addr;
8    char            *ai_canonname;
9    struct addrinfo *ai_next;
10 };
```

zob. `getaddrinfo(3)`

</> Połączenie TCP IPv4/IPv6

```
1  int _connect(const char *host, const char *service) {
2    int sfd;  struct addrinfo hints, *res, *ressave;
3    memset(&hints, 0, sizeof(hints));
4    hints.ai_family = AF_UNSPEC;
5    hints.ai_socktype = SOCK_STREAM;
6    getaddrinfo(host, service, &hints, &res);
7    ressave = res;
8    do {
9      sfd = socket(res->ai_family, res->ai_socktype,
10                  res->ai_protocol);
11     if (connect(sfd, res->ai_addr, res->ai_addrlen) == 0)
12       break;
13     close(sfd);
14   } while((res = res->ai_next) != NULL);
15   freeaddrinfo(ressave);  return sfd;
16 }
```


Podsumowanie

Podsumowanie

- Protokół IPv6:
  - budowa i zapis adresów;
  - typy adresów;
  - tunelowanie IPv6 w IPv4.
- Protokół IPv6 w systemach operacyjnych GNU/Linux.
- Gniazda sieciowe protkołu IPv6.
- Implementacje zależne od protokołu warstwy sieciowej.
- Implementacje niezależne od protokołu warstwy sieciowej.

Pytania i zadania

Pytania i zadania — praca własna

1. Jakie są adresy *specjalne* protokołu IPv6?
2. Czy zastosowanie protokołu IPv6 w miejsce protokołu IPv4 wpływa na efektywność protokołów transportowych?
3. Dowiedz się jak dokładnie działa protokół *Neighbor Discovery Protocol* (NDP).
4. Sprawdź, czy popularne aplikacje sieciowe (np. przeglądarki internetowe) wspierają komunikację z użyciem protokołu IPv6.

Dziękuję za uwagę!