Laboratorium 11

Programowanie sieciowe 2

Maryna Lukachyk(308294)

1 Komunikacja bezpołączeniowa

💡 **W jakich przypadkach stosuje się komunikację bezpołączeniową z użyciem UDP?**

User Datagram Protocol (UDP) został stworzony aby móc przesyłać datagramy pomiędzy komputerami podłączonymi do sieci komunikacyjnej.

Gdy zależy nam na maksymalnej wydajności i tolerujemy utratę danych (albo sami implementujemy niezawodność w warstwie wyższej!)

Nie podlegamy automatycznemu ograniczaniu prędkości (dobrze dla nas, źle dla Internetu ...)

Odpowiednie dla: strumieniowe multimedia, telefonia VOIP, routing, DNS, sieci P2P.

💡 Czy programy korzystające z protokołu UDP mają możliwość kontroli pakietów? Podaj przykład, jak mogą to kontrolować (np. jak to jest rozwiązane w przypadku programu tftp)

Tak, mają możliwość kontroli pakietów.

W przypadku aplikacji wykorzystujących ten właśnie protokół toleruje się to, że czasem jakiś pakiet może zostać utracony, bądź uszkodzony. W przypadku usługi DNS, jeśli datagram się zgubi to po prostu zostaje jeszcze raz wysłane zapytanie do serwera DNS, jeśli podczas telekonferencji jakiś datagram nie dotrze to też nie będzie tragedii, bo zawsze komunikat można powtórzyć. W przypadku aplikacji korzystających z TCP utrata czy zagubienie akceptowalne już nie jest. Datagramy odbierane są w takiej kolejności w jakiej zostały

odebrane, a jeśli jest ich dużo, to za ich odpowiednie poskładanie odpowiada już konkretna aplikacja.

Listener.c

int sockfd = socket(domain, type, protocol)

sockfd:

socket descriptor, an integer (like a file-handle)

domain:

integer, communication domain e.g., AF_INET (IPv4 protocol) , AF_INET6 (IPv6 protocol)

type:

communication type

SOCK_STREAM: TCP(reliable, connection oriented)

SOCK_DGRAM: UDP(unreliable, connectionless)

protocol:

Protocol value for Internet Protocol(IP), which is 0. This is the same number which appears on protocol field in the IP header of a packet.(man protocols for more details)

```
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
    perror("socket");
    exit(1);
}

my_addr.sin_family = AF_INET;;
my_addr.sin_port = htons(MYPORT);
my_addr.sin_addr.s_addr = INADDR_ANY; // automatically fill with my IP
memset(my_addr.sin_zero, '\0', sizeof my_addr.sin_zero);

if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof my_addr) == -1) {
    perror("bind");
    exit(1);
}
```

talker.c

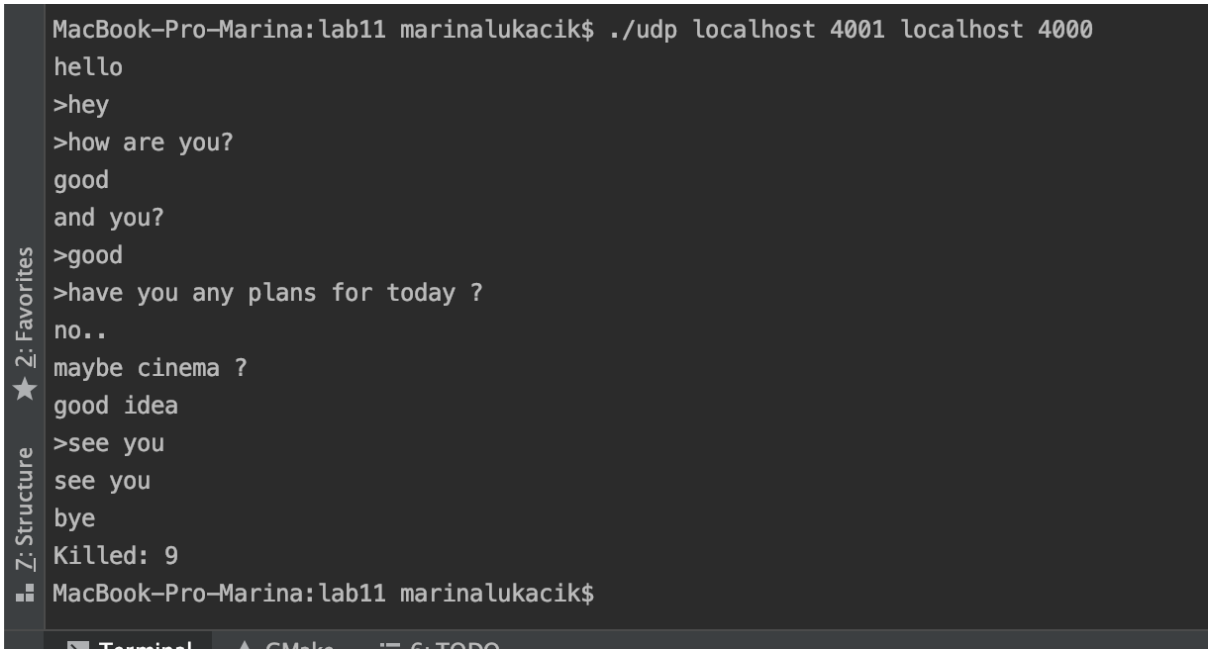
```
int sendto(int sockfd, const void *msg, int len, unsigned int flags,
           const struct sockaddr *to, socklen_t tolen);
```

```c
if ((numbytes = sendto(sockfd, argv[2], strlen(argv[2]), 0,
                       (struct sockaddr *)&their_addr, sizeof their_addr)) == -1) {
    perror("sendto");
    exit(1);
}
```

Jak widać, w komunikacji z użyciem protokołu UDP serwer nie różni się od klienta tak jak w przypadku korzystania z TCP. Można zatem użyć tylko jednego programu i komunikować sie na zasadzie peer-to-peer.

udpchat.c

```
MacBook-Pro-Marina:lab11 marinalukacik$ ./udp localhost 4000 localhost 4001
>hello
hey
how are you?
>good
>and you?
good
have you any plans for today ?
>no..
>maybe cinema ?
>good idea
see you
>see you
>bye
Killed: 9
MacBook-Pro-Marina:lab11 marinalukacik$
```



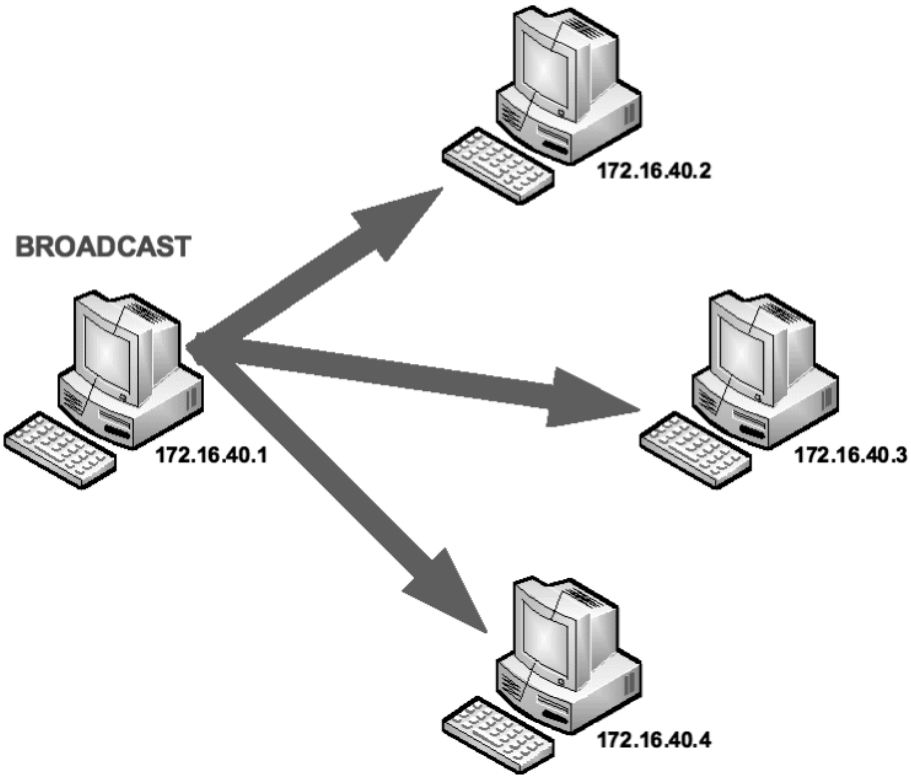
2 Broadcasting

Transmisja broadcast polega na wysyłaniu pakietów przez jeden port (kanał komunikacyjny), które powinny odbierać wszystkie pozostałe porty przyłączone do danej sieci (domeny rozgłoszeniowej).

Pakiet danych, wysyłany do wszystkich stacji sieciowych domeny rozsiewczej, ma adres składający się z samych jedynek.(1.1.1.1)

255.255.255.255 – adres tego typu jest stosowany w wiadomości wysłanej do wszystkich urządzeń i wszystkich sieci (podsieci). Wiadomość taka byłaby niebezpieczna dla funkcjonowania Internetu i dlatego routery nie przełączają takiego pakietu, co ogranicza jego rozprzestrzenianie jedynie do sieci lokalnej.

Transmisja broadcast



Funkcje `setsockopt()` i `getsockopt()`

Za pomoca tych funkcji mozemy manipulowac zarowno opcjami dotyczacymi ogolnego programu obslugi gniazd, jak i poszczegolnych warstw protokolu w obrebie ktorego dziala gniazdo.

```
 DEFINCJE: int setsockopt (int sockfd, int level, int optname, char *optval, int optlen)
           int getsockopt (int sockfd, int level, int optname, char *optval, int *optlen)
    WYNIK:  0 w przypadku powodzenia
           -1 gdy blad:
                 errno = EBADF       (argument sockfd nie jest poprawnym deskryptorem)
                         ENOTSOCK    (sockfd nie jest deskryptorem gniazda)
                         ENOPROTOOPT (nieprawidlowa opcja na wskazanym poziomie)
                         EFAULT      (optval wskazuje na nieprawidlowy adres)
```

Znaczenie argumentow jest nastepujace:

`sockfd` - deskryptor gniazda, `level` - poziom, ktorego dotyczy opcja, mozliwe wartosci: **`SOL_SOCKET`** - nawyzszy poziom - ogolny program obslugi gniazd, **`IPPROTO_xxx`** - opcje protokolow,

`optval` - wskaznik zmiennej uzytkownika, z ktorej pobierana jest wartosc ustanawianej opcji (setsockopt) lub na ktora zapisuje sie wartosc wybranej opcji (getsockopt),

`optlen` - rozmiar zmiennej wskazywanej przez optval.