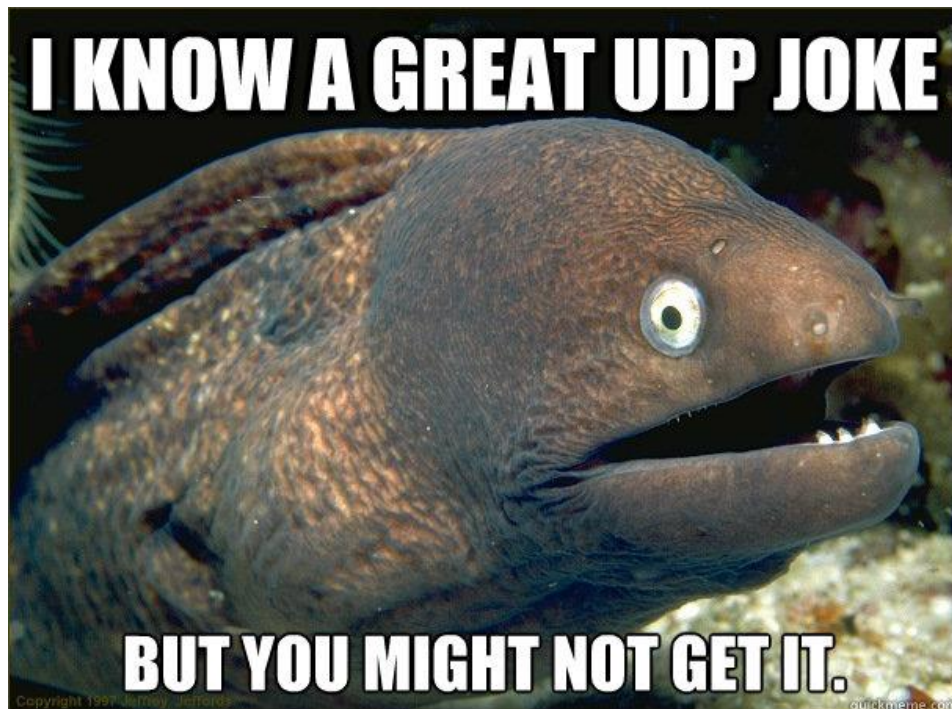


## Øvelse 9 - UDP



20105969 - Kalle R. Møller

201370050 - Kasper Sejer Kristensen

## UDP Socket Programming

I denne øvelse har vi fået til opgave at programmere et UDP server/client setup der er i stand til at sende tekst strenge frem og tilbage

### Protokollen

Vi er har specificeret protokollen således:

#### Beskedformat:

Klienten sender: Kommando

Kommandoer

- L : Load på serveren
- U : Uptime på serveren

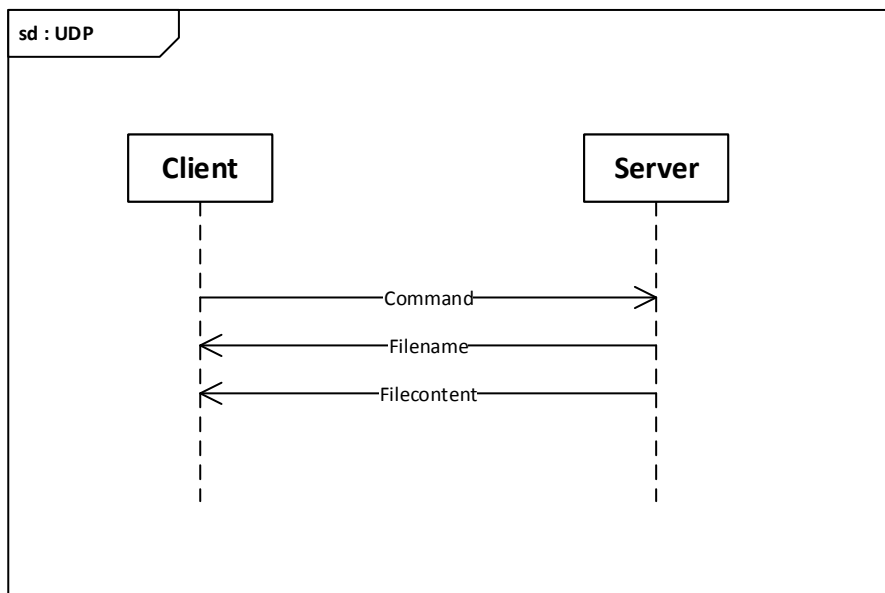
Server svarer med to strenge

- s1: Navnet på filen der blev brugt
- s2: Indholdet af denne file

U kommandoen mapper til /proc/uptime

L kommandoen mapper til /proc/loadavg

Sekvensen ser således ud



## UDP Sockets

Her beskrives kort hvordan vi har programmeret med UDP sockets. Der bliver fremhævet de vigtigste ting i koden der gør at det virker både for server og for klienten.

## Server

Serveren er skrevet i c# (mono). Socket biblioteket er blevet brugt til at oprette en UDP socket som serveren bruger til at lytte efter klienter med.

```
Socket JustSocket = new Socket(
    AddressFamily.InterNetwork,
    SocketType.Dgram,
    ProtocolType.Udp
);
private int port = 9999;

<----->

IPEndPoint ep = new IPEndPoint(IPAddress.Any, port);
JustSocket.Bind(ep);
```

I den viste kode kan det ses hvordan der oprettes en socket med parametrene InterNetwork, Dgram og Udp de fortæller at vi ville have en UDP socket der kommunikerer over IP. Efter at vi har oprettet en socket binder vi til alle indkommende IP'er på port 9999. Til slut kaldes Bind() som gør at vi lytter efter klienter der vil i kontakt med os.

```
received = JustSocket.ReceiveFrom(data, ref Remote);
UdpWorker work = new UdpWorker(Remote, Encoding.UTF8.GetChars(data));
Thread WorkerThread = new Thread(work.Run);
WorkerThread.Start();
```

Efter at socket er oprettet og vi er sat til at lytte sættes serveren til at stå i en while(1) løkke hvor metode i ovenstående kode bliver kaldt den modtager så de data der er sendt, og putter dem in i en ny tråd sammen med afsenderen. Denne sørger så for at svare på requestet, samt vi kan modtage en ny.

Når UdpWorker'en bliver kørt med klientens request, tester den med en switch case om hvilken kommando der er tale om og svarer via

```
responseSocket.SendTo(
    Encoding.UTF8.GetBytes(response_str),
    response_str.Length,
    SocketFlags.None,
    _remote
);
```

Hvor argumenterne er selve strengen udtrykt i bytes, længden, socket flags (her ingen) samt hvilken remote der skal svares til.

## Klienten

For udfordringens skyld blev klienten skrevet i et andet sprog nemlig c det er ingen umiddelbar grund til dette og det har ingen betydning for funktionaliteten for c# serveren.

I c findes der en biblioteket der hedder socket.h denne bliver brugt til at oprette og kommunikere med serveren.

```
s = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
```

Som i serveren bruger vi her

AF\_INET, SOCK\_DGRAM og IPPROTO\_UDP

Dette er for at vælge UDP datagrams over IP

Vi bruger følgende til at klargøre modtagelsen af svaret fra serveren.

```
memset((char *) &udpserv, 0, len);
udpserv.sin_family = AF_INET;
udpserv.sin_port = htons(PORT);
udpserv.sin_addr = *((struct in_addr *) host->h_addr);
```

Vi allokerer memory, samt sætter indstillingerne angående svarport og hvilke ip'er vi lytter på

Når vi er klar til at modtage svaret fra serveren sender vi vores kommando

```
sendto(
    s,
    argv[CMD_ARG],
    strlen(argv[CMD_ARG]),
    0,
    (struct sockaddr *) &udpserv,
    len
);
```

Igen er det selve kommandoen, længden, flags(her ingen), samt serveren

Vi venter derefter på svar fra serveren i et while loop

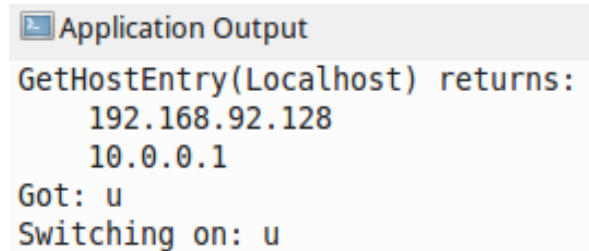
```
n = recvfrom(s, buf, BUFFER_SIZE, 0, (struct sockaddr *) &udpserv, &len)
```

Som vi printer

```
printf("Modtaget %s:%d: ", inet_ntoa(udpserv.sin_addr), ntohs(udpserv.sin_port));
```

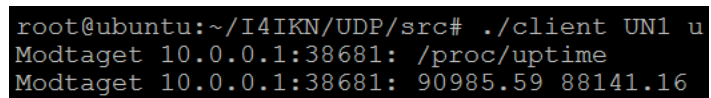
## Resultater

Herunder vises et screenshot fra en mono hvor serveren kører den har netop fået en besked og sender sin uptime tilbage:



```
Application Output
GetHostEntry(localhost) returns:
    192.168.92.128
    10.0.0.1
Got: u
Switching on: u
```

Her ses klienten der sendte requesten



```
root@ubuntu:~/I4IKN/UDP/src# ./client UN1 u
Modtaget 10.0.0.1:38681: /proc/uptime
Modtaget 10.0.0.1:38681: 90985.59 88141.16
```