UDP sockets

Veerle Ongenae

UDP functie-overzicht

- client
  - socket(DATAGRAM)
  - sendto()
  - recvfrom()
  - close()
- server
  - socket(DATAGRAM)
  - bind()
  - recvfrom
  - sendto

recvfrom( )

```
int recvfrom(
  int sockfd, // socket descriptor
  void *buf, // bericht
  int len, // lengte bericht
  int flags,  // opties
  struct sockaddr *from, //adres verzender
  int *fromlen); // adreslengte
```

Return : # ontvangen bytes, -1 error

sendto( )

```
int  sendto(
 int sockfd,  // socketdescriptor
 const void *msg, // bericht
 int len, // lengte bericht
 int flags, // opties
 const struct sockaddr *to, // naar waar?
 int tolen); // lengte adres

Return # bytes, -1 bij error
```

Opgelet !

- datagram met lengte 0 is ok ◊ functies geven 0 terug
- je kan dus datagrammen met lengte 0 versturen en ontvangen
- 0 voor “einde connectie” immers irrelevant bij UDP

UDP Server code

- UDPEchoServer.cpp

UDP Servers

- UDP servers zijn iteratief maar wel verschillende clients mogelijk aangezien er geen connecties zijn

UDP Echo client

- UDPEchoClient.cpp

Betrouwbaarheid

- de echo client is onbetrouwbaar; indien een datagram verloren gaat blokkeert hij eeuwig in aanroep recvfrom()
- oplossing : met select() een time-out opgeven

Betrouwbaarheid

- UDPEchoClientSelect.cpp

Betrouwbaarheid

- Willekeurig proces kan een datagram sturen
  - Controleren serveradres

```
void do_echo( int fd, int sockfd, const sockaddr
  *pservaddr, int servlen){

  int n, len;
  char buffer[MAXLINE+1];
  struct sockaddr *preply_addr;
  preply_addr = malloc(servlen);

  while (read(fd, buffer, MAXLINE) > 0) {
    sendto( sockfd, buffer, strlen(buffer), 0,
     pservaddr, servlen);
    len = servlen;
    n = recvfrom(sockfd, buffer, MAXLINE, 0,
     preply_addr, &len);
    if (len != servlen || memcmp(pservaddr, preply_addr,
  len) != 0) {
      cout << "antwoord genegeerd" << endl;
    }
    else{
      buffer[n] = 0;    /* null terminate */
      cout << buffer;
    }
  }
```