

Introdução às Redes de Comunicação

Ficha de exercícios (UDP)

1. Desenvolva uma aplicação cliente-servidor em que os clientes enviam, via protocolo UDP, uma mensagem de texto, passada como argumento através da linha de comando, para um servidor específico com localização dada pelas constantes *SERV_HOST_ADDR* (endereço IP) e *SERV_UDP_PORT* (porto). O servidor deve ir apresentado na saída *standard* as mensagens que vai recebendo.
2. Altere a aplicação anterior de modo a que o servidor reenvie as mensagens recebidas aos respectivos clientes. Estes devem aguardar pelas respostas e apresentá-las na saída *standard*.
3. Altere o cliente do exercício anterior de modo a que este apresente, na saída *standard*, o porto local automaticamente atribuído ao seu *socket* UDP. Esta operação deve ser executada depois do envio da mensagem ao servidor.
4. Altere o servidor desenvolvido no exercício 2 de modo a que este mostre, além dos conteúdos, as localizações de origem das mensagens recebidas.
5. Altere o cliente do exercício 3 de modo a que a localização do servidor seja fornecida através de argumentos passados na linha de comando. Para flexibilizar a ordem pela qual os argumentos são fornecidos, pode recorrer a uma sintaxe baseada em *flags* (por exemplo, -msg “Este exercício é ultra simples” -ip 127.0.0.1 -port 5001).
6. Altere o cliente do exercício anterior de modo a que este verifique se a resposta recebida foi efectivamente enviada pelo servidor.
7. Ponha a correr o servidor desenvolvido em todos os computadores do laboratório, usando o porto de escuta 5001. Invoque, repetidamente, o cliente fornecendo como endereço ip e

porto de destino os valores “255.255.255.255” e 5001, respectivamente. Observe e interprete aquilo que acontece.

8. Altere o cliente desenvolvido ao longo dos últimos exercícios de modo a que este aguarde pela resposta do servidor apenas durante um determinado tempo especificado pela constante *TIMEOUT*.
9. Altere o cliente e o servidor que foram desenvolvidos ao longo dos exercícios anteriores de modo a que a resposta do servidor contenha o tamanho da mensagem recebida em formato *ascii*.
10. Altere o cliente e o servidor que foram desenvolvidos ao longo dos exercícios anteriores de modo a que a resposta do servidor contenha o tamanho da mensagem recebida em formato binário. Preveja a possibilidade de ambientes distribuídos heterogéneos.
11. Desenvolva uma aplicação cliente-servidor que obedeça aos seguintes requisitos. O servidor serve de “casamenteiro” entre pares de clientes UDP. Ao receber uma mensagem (o conteúdo é irrelevante) do primeiro cliente (c1), o servidor anota o endereço do mesmo numa estrutura do tipo *struct sockaddr_in*, mas não lhe responde. Quando receber uma mensagem de um segundo cliente (c2), o servidor remete o endereço de c1 a c2, sob a forma de uma variável do tipo *struct sockaddr_in* (em formato binário), dando por concluída a sua tarefa relativamente a este par. O servidor passa, então, a aguardar pelo próximo par de clientes que o contactarem. Depois de enviar uma mensagem ao servidor, qualquer cliente aguarda por uma resposta cujo conteúdo esperado é uma variável do tipo *struct sockaddr_in*. Se a resposta tiver sido enviada pelo servidor, o cliente (i.e., c2) interpreta o seu conteúdo como sendo a localização do seu par (i.e., c1) e reencaminha-a para este. Se o endereço recebido não tiver sido enviado pelo servidor, o cliente (i.e., c1) assume que está a ser contactado pelo seu par (i.e., c2). Em ambas as situações os clientes reportam a localização da sua “cara-metade” na *saída standard* e terminam, dando-se o “nó” por concluído.

CRIAÇÃO, ASSOCIAÇÃO A UM PORTO LOCAL E FECHO DE SOCKETS WINDOWS

```
SOCKET socket(int af, int type, int protocol); /* PF_INET, SOCK_DGRAM, IPPROTO_UDP */  
  
int bind(SOCKET s, const struct sockaddr *name, int namelen);  
  
int closesocket(SOCKET s);
```

INDICAÇÃO DE ERRO E CÓDIGOS DE ERRO

```
SOCKET_ERROR  
  
INVALID_SOCKET  
  
int WSAGetLastError(void); /* WSAETIMEDOUT */
```

LOCALIZAÇÃO E CONVERSÃO DE FORMATOS

```
struct sockaddr_in a; /* a.sin_family, a.sin_addr.s_addr, a.sin_port */  
  
...htons(...); /* host to network short */  
  
...htonl(...); /* host to network long */  
  
...ntohs(...); /* network to host short */  
  
...ntohl(...); /* network to host long */  
  
unsigned long inet_addr(const char *cp);  
  
char* inet_ntoa(struct in_addr in); /* network to ascii */
```

ENVIO E RECEPÇÃO DE DATAGRAMAS

```
int sendto(SOCKET s, const char *buf, int len, int flags, struct sockaddr *to, int tolen);  
  
int recvfrom(SOCKET s, char *buf, int len, int flags, struct sockaddr *from, int *fromlen);
```

OBTENÇÃO DE INFORMAÇÃO LOCAL ASSOCIADA AOS SOCKETS

```
int getsockname(SOCKET s, struct sockaddr *name, int *namelen);  
  
int strcmp(const char *s1, const char *s2);  
  
char * strcpy_s(char * strDestination, int sizeStrDestination, const char * strSource);
```

CONFIGURAÇÃO DE OPÇÕES/PARÂMETROS

```
int setsockopt(SOCKET s, int level, int optname, const char *optval, int optlen);  
  
/* level = SOL_SOCKET, optname = SO_RCVTIMEO, optval = (char *)&timeoutMsec (DWORD **  
timeoutMsec;) */
```