INF01018
– Aula Prática 1 –
Cliente-Servidor e Multicast

Lucas Mello Schnorr, Alexandre Silva Caríssimi

{lmschnorr,asc}@inf.ufrgs.br

http://www.inf.ufrgs.br/∼lmschnorr/ad/

INF01018 – Sistemas Operacionais
Distribuídos e de Redes

Sala 102 – Prédio 67 – 05 Setembro 2006

Roteiro

Introdução

- Dar suporte às aulas teóricas 04 e 05
- Utilização de exemplos com código fonte
- Duas linguagens de programação: Java e C
- Programas Cliente/Servidor e Multicast

Download de Exemplos

http://www.inf.ufrgs.br/∼lmschnorr/ad/aula1.tar.gz

Introdução – Parte do Cliente/Servidor

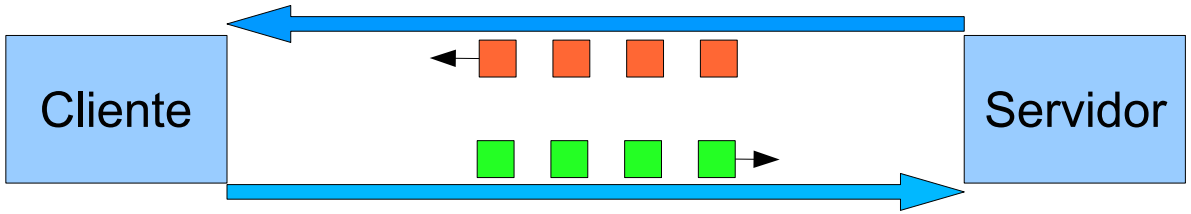
- Do lado do servidor
  - Recebe comandos dos clientes
  - Se comando for igual a hora, enviar hora local ao cliente
  - Se comando não for conhecido, envia mensagem ao cliente avisando
- Do lado do cliente
  - Cria uma datagrama com o comando
  - Envia ao servidor
  - Aguarda uma resposta

Introdução – Parte do Multicast

- Dois programas
- Programa Receptor
  - Cria um socket multicast
  - Faz bind desse socket a um endereço multicast
  - Aguarda mensagens do socket e imprime na tela
- Programa Enviador
  - Cria um socket multicast
  - Entra no grupo multicast
  - Envia uma mensagem ao grupo

Cliente-Servidor

- UDP
  - Serviço orientado a Datagramas
  - Sem confirmação de recebimento ou retentativas



- Do lado do servidor
  - Recebe comandos dos clientes
  - Se comando for igual a hora, enviar hora local ao cliente
  - Se comando não for conhecido, envia mensagem ao cliente avisando
- Do lado do cliente
  - Cria uma datagrama com o comando
  - Envia ao servidor
  - Aguarda uma resposta

UDP usando a linguagem C

Principais funções

socket (int domain, int type, int protocol**)**

bind (int sockfd, struct sockaddr *addr, socklen_t len**)**

recvfrom (int s, void *buf, size_t len, int flags,
struct sockaddr *from, socklen_t *fromlen**)**

sendto (int s, const void *buf, size_t len, int flags,
const struct sockaddr *to, socklen_t tolen**)**

Cliente-Servidor UDP em C – 1/3

Códigos retirados do arquivo ServidorUDP.c

```
30      /* criar socket UDP para receber ou enviar datagramas */
31      if ((sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0){
32              perror ("sock() falhou");
33      }
```

```
35      /* construir a estrutura de endereco local */
36      memset(&echoServAddr, 0, sizeof(echoServAddr));
37      echoServAddr.sin_family = AF_INET;
38      echoServAddr.sin_addr.s_addr = htonl(INADDR_ANY);
39      echoServAddr.sin_port = htons(numeroPorta);
```

```
41      /* Bind ao endereco local */
42      if (bind(sock, (struct sockaddr *) &echoServAddr,
43                                      sizeof(echoServAddr)) <0){
44              perror ("bind() falhou");
45      }
```

Cliente-Servidor UDP em C – 2/3

Códigos retirados do arquivo ServidorUDP.c

```
47      while (1){
48          /* esperar mensagem de alguem */
49          cliAddrLen = sizeof(echoClntAddr);
50          if ((recvMsgSize = recvfrom(sock, echoBuffer, ECHOMAX, 0,
51              (struct sockaddr *) &echoClntAddr, &cliAddrLen)) <0){
52                  perror ("recvfrom () falhou");
53          }
54          printf ("Recebeu cliente %s\n",
55                          inet_ntoa (echoClntAddr.sin_addr));
56          printf ("==> [%s] %d\n", echoBuffer, cliAddrLen);
```

```
68                  /* enviar a HORA para o cliente */
69                  int bytesEnviados = sendto(sock, time_string,
70                          sizeof(time_string), 0,
71                          (struct sockaddr *)&echoClntAddr,
72                          sizeof(echoClntAddr));
73                  printf ("enviando %d\n", bytesEnviados);
```

Cliente-Servidor UDP em C – 3/3

Códigos retirados do arquivo ClienteUDP.c

```
36      /* criar um socket UDP */
37      if ((sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0){
38              perror ("socket() falhou");
39      }
```

```
41      /* construir a estrutura de endereco do servidor*/
42      memset(&echoServAddr, 0, sizeof(echoServAddr));
43      echoServAddr.sin_family = AF_INET;
44      echoServAddr.sin_addr.s_addr = inet_addr(nomeServidor);
45      echoServAddr.sin_port    = htons(numeroPorta);
```

```
47      /* enviar o comando para o servidor */
48      sendto(sock, mensagem, tamanhoMensagem+1, 0,
49                  (struct sockaddr *) &echoServAddr,
50                  sizeof(echoServAddr));
```

```
52      /* receber uma resposta */
53      fromSize = sizeof(fromAddr);
54      respStringLen = recvfrom(sock, echoBuffer, ECHOMAX, 0,
55           (struct sockaddr *) &fromAddr, &fromSize);
```

UDP usando a linguagem Java

Classes utilizadas

DatagramSocket

new DatagramSocket (int porta)

DatagramPacket

new DatagramPacket (byte[] b, int length)

new DatagramPacket (byte[] buf, int length,
InetAddress address, int port)

Cliente-Servidor UDP em Java – 1/3

Códigos retirados do arquivo ServidorUDP.java

```
18      /* Inicializacao do socket UDP */
19      socket = new DatagramSocket(
20          new Integer(numeroPorta).intValue());
```

```
22      /* Laco de recebimento de datagramas */
23      while (true){
24        request = null;
25        reply = null;
26        buf = new byte[1024];
27
28        /* Preparacao do Datagrama de Recepcao */
29        request = new DatagramPacket (buf, buf.length);
```

```
31        /* Recepcao bloqueante dos dados */
32        socket.receive (request);
33
34        /* Recuperacao do comando */
35        comando = new String (request.getData(),0,
36            request.getLength());
```

Cliente-Servidor UDP em Java – 2/3

Códigos retirados do arquivo ServidorUDP.java

```
43          /* Cria datagrama com a resposta */
44          reply = new DatagramPacket (hora.getBytes(),
45              hora.getBytes().length,
46              request.getAddress(),
47              request.getPort());
```

```
56          /* Envia resposta pelo socket UDP */
57          socket.send (reply);
```

Cliente-Servidor UDP em Java – 3/3

Códigos retirados do arquivo ClienteUDP.java

```
20      /* Inicializacao de sockets UDP com Datagrama */
21      socket = new DatagramSocket();
```

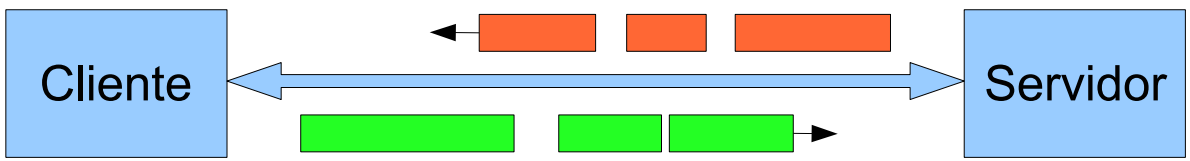
```
23      /* Configuracao a partir dos parametros */
24      InetAddress host = InetAddress.getByName(nomeServidor);
25      serverPort = new Integer (numeroPorta).intValue();
26      byte[] m = mensagemEnviar.getBytes();
27
28      /* Criacao do Pacote Datagrama para Envio */
29      request = new DatagramPacket(m, m.length , host , serverPort);
```

```
31      /* Envio propriamente dito */
32      socket.send (request);
```

```
34      /* Preparacao do Pacote Datagrama para Recepcao */
35      reply = new DatagramPacket(buf , buf.length);
36
37      /* Recepcao do retorno */
38      socket.receive (reply);
```

Cliente-Servidor em TCP

- TCP
  - Orientado a conexão (stream)
  - Confiabilidade e Integridade
  - Menos desempenho na comunicação



TCP usando a linguagem C

Principais funções

socket (int domain, int type, int protocol)

bind (int sockfd, struct sockaddr *addr, socklen_t len)

listen (int sockfd, int backlog)

accept (int sockfd, struct sockaddr *addr, socklen_t *addrlen);

send (int s, const void *buf, size_t len, int flags);

recv (int s, void *buf, size_t len, int flags);

connect (int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);

Cliente-Servidor TCP em C – 1/4

Códigos retirados do arquivo ServidorTCP.c

```
36      /* Criacao do socket TCP para receber conexoes */
37      if ((servSock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP))<0){
38              perror ("socket() falhou");
39              exit(1);
40      }
```

```
43      /* construcao do endereco de conexao */
44      memset(&echoServAddr, 0, sizeof(echoServAddr));
45      echoServAddr.sin_family = AF_INET;
46      echoServAddr.sin_addr.s_addr = htonl(INADDR_ANY);
47      echoServAddr.sin_port = htons(numeroPorta);
```

```
49      /* Bind ao endereco local */
50      if (bind(servSock, (struct sockaddr *) &echoServAddr,
51                              sizeof(echoServAddr)) < 0){
52              perror ("bind() falhou");
53              exit(1);
54      }
```

Cliente-Servidor TCP em C – 2/4

Códigos retirados do arquivo ServidorTCP.c

```
57      /* listen */
58      if (listen(servSock, MAXPENDING) < 0){
59              perror ("listen() falhou");
60              exit(1);
61      }
```

```
64      while (1){
65          /* esperando conexoes com accept */
66          clntLen = sizeof(echoClntAddr);
67          if ((clntSock = accept(servSock,
68                          (struct sockaddr *) &echoClntAddr,
69                                  &clntLen)) < 0){
70                  perror ("accept() falhou");
71                  exit(1);
72          }
```

Cliente-Servidor TCP em C – 3/4

Códigos retirados do arquivo ServidorTCP.c

```
74          /* clntSock esta conectado a um cliente */
75          printf("Tratando cliente %s\n",
76                              inet_ntoa(echoClntAddr.sin_addr));
77          while (1){
78                  if ((tamanhoRecebido = recv (clntSock, comando,
79                                               RCVBUFSIZE, 0)) < 0){
80                          perror ("recv() falhou\n");
81                          exit(1);
82                  }
```

```
94          /* enviando dados ao cliente */
95          int k = send (clntSock, time_string,sizeof(time_string),0);
```

Cliente-Servidor TCP em C – 4/4

Códigos retirados do arquivo ClienteTCP.c

```
34      /* Criacao do Socket TCP */
35      sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
```

```
37      /* construcao do endereco de conexao */
38      memset(&echoServAddr, 0, sizeof(echoServAddr));
39      echoServAddr.sin_family = AF_INET;
40      echoServAddr.sin_addr.s_addr = inet_addr(enderecoIPServidor);
41      echoServAddr.sin_port = htons(numeroPorta);
```

```
43      /* Conectando ao servidor */
44      if (connect(sock, (struct sockaddr *) &echoServAddr,
45                                  sizeof(echoServAddr)) < 0){
46          perror ("connect() falhou:");
47          exit(1);
48      }
```

TCP usando a linguagem Java

Principais Classes

ServerSocket (int port)

PrintWriter (OutputStream out, boolean autoFlush)

BufferedReader (Reader in)

Cliente-Servidor TCP em Java – 1/2

Códigos retirados do arquivo ServidorTCP.java

```
20      /* Inicializacao do server socket TCP */
21      serverSocket = new ServerSocket(
22          new Integer (numeroPorta).intValue());
```

```
24      while (true){
25        /* Espera por um cliente */
26        clientSocket = serverSocket.accept();
```

```
29        /* Preparacao dos fluxos de entrada e saida */
30        out = new PrintWriter(clientSocket.getOutputStream(),
31            true);
32        in = new BufferedReader(new InputStreamReader(
33            clientSocket.getInputStream()));
```

```
35        /* Recuperacao dos comandos */
36        while ((comando = in.readLine()) != null) {
```

```
42            /* Escreve na saida a 'hora' */
43            out.println (hora);
```

Cliente-Servidor TCP em Java – 2/2

Códigos retirados do arquivo ClienteTCP.java

```
17      /* Inicializacao de socket TCP */
18      socket = new Socket(nomeServidor,
19          new Integer(numeroPorta).intValue());
```

```
21      /* Inicializacao dos fluxos de entrada e saida */
22      in = new BufferedReader(new InputStreamReader(
23              socket.getInputStream()));
24      out = new PrintWriter(socket.getOutputStream(), true);
```

```
30      while ((mensagemEnviar = inReader.readLine()) != null){
31
32        /* Envio da mensagem */
33        out.println (mensagemEnviar);
34
35        /* Recebimento da resposta do servidor */
36        String resposta = in.readLine ();
```

```
45      out.close();
46      in.close();
47      socket.close();
```

Multicast

- Utilizado com suporte da camada Ethernet
- Quando sem esse suporte, utiliza vários Unicast
- Conceito de grupo

Multicast usando a linguagem C

Principais funções

setsockopt (int s, int level, int optname,
const void *optval, socklen_t optlen);

Multicast em C – 1/3

Códigos retirados do arquivo MulticastReceiver.c

```
28      if ((sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0){
29              printf ("socket() falhou\n");
30      }
```

```
32      /* construir a estrutura de endereco para o bind */
33      memset(&multicastAddr, 0, sizeof(multicastAddr));
34      multicastAddr.sin_family = AF_INET;
35      multicastAddr.sin_addr.s_addr = htonl(INADDR_ANY);
36      multicastAddr.sin_port = htons(multicastPort);
```

```
38      /* bind */
39      if (bind(sock, (struct sockaddr *) &multicastAddr,
40                                      sizeof(multicastAddr)) < 0){
41              printf ("bind() falhou\n");
42      }
```

Multicast em C – 2/3

Códigos retirados do arquivo MulticastReceiver.c

```
44      /* acertar opcoes do multicast */
45      multicastRequest.imr_multiaddr.s_addr =
46              inet_addr(multicastIP);
47      multicastRequest.imr_interface.s_addr = htonl(INADDR_ANY);
48      if (setsockopt(sock, IPPROTO_IP, IP_ADD_MEMBERSHIP,
49                              (void *) &multicastRequest,
50                              sizeof(multicastRequest)) < 0){
51              printf ("setsockopt() falhou\n");
52      }
```

```
54      /* Receive a single datagram from the server */
55      while (1){
56              if ((recvStringLen = recvfrom (sock, recvString,
57                              MAXRECVSTRING, 0, NULL, 0)) < 0){
58                      printf ("recvfrom() falhou\n");
59              }
```

Multicast em C – 3/3

Códigos retirados do arquivo MulticastSender.c

```
30      if ((sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0){
31          printf ("socket() falhou\n");
32      }
```

```
39      memset (&multicastAddr, 0, sizeof(multicastAddr));
40      multicastAddr.sin_family = AF_INET;
41      multicastAddr.sin_addr.s_addr = inet_addr (multicastIP);
42      multicastAddr.sin_port = htons (multicastPort);
```

```
50              int k = sendto (sock, sendString, sendStringLen, 0,
51                      (struct sockaddr *) &multicastAddr,
52                      sizeof(multicastAddr));
```

Multicast usando a linguagem Java

Principais classes

MulticastSocket (int port)

DatagramPacket (byte[] buf, int length)

Multicast em Java – 1/3

Códigos retirados do arquivo ServidorMulticast.java

```
11          MulticastSocket socket = new MulticastSocket (porta);

12          InetAddress endereco = InetAddress.getByName (args[1]);
13          socket.joinGroup (endereco);

15          while(true) {
16                  byte[] recvData = new byte[1024];
17                  DatagramPacket recvPacket;
18                  recvPacket = new DatagramPacket (recvData,
19                                  recvData.length);
20                  socket.receive (recvPacket);

23                  sentence = new String (recvPacket.getData ());
24                  System.out.print
25                          (recvPacket.getAddress().toString() + ": ");
26                  System.out.println (sentence);
```

Multicast em Java – 1/3

Códigos retirados do arquivo ClienteMulticast.java

```
13      MulticastSocket clientSocket = new MulticastSocket();
```

```
14      InetAddress endereco = InetAddress.getByName(args[1]);
15      clientSocket.joinGroup(endereco);
```

```
26        sendPacket = new DatagramPacket (sendData,
27            sendData.length, endereco, porta);
28        clientSocket.setTimeToLive (ttl);
29        clientSocket.send (sendPacket);
```

```
34      clientSocket.leaveGroup(endereco);
35      clientSocket.close();
```

Referências

- Man pages no Linux para funções C
- Especificação das classes Java na Internet