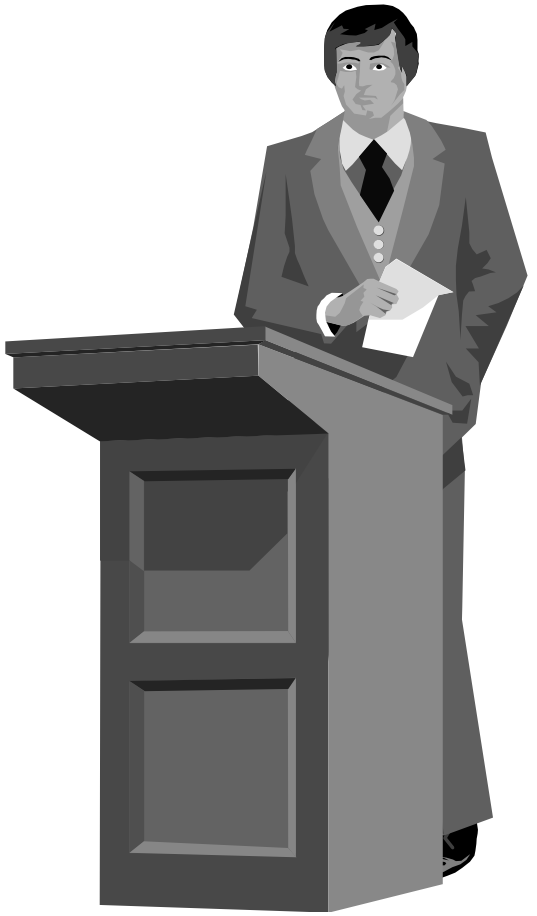
Trabalho
Chat UDP

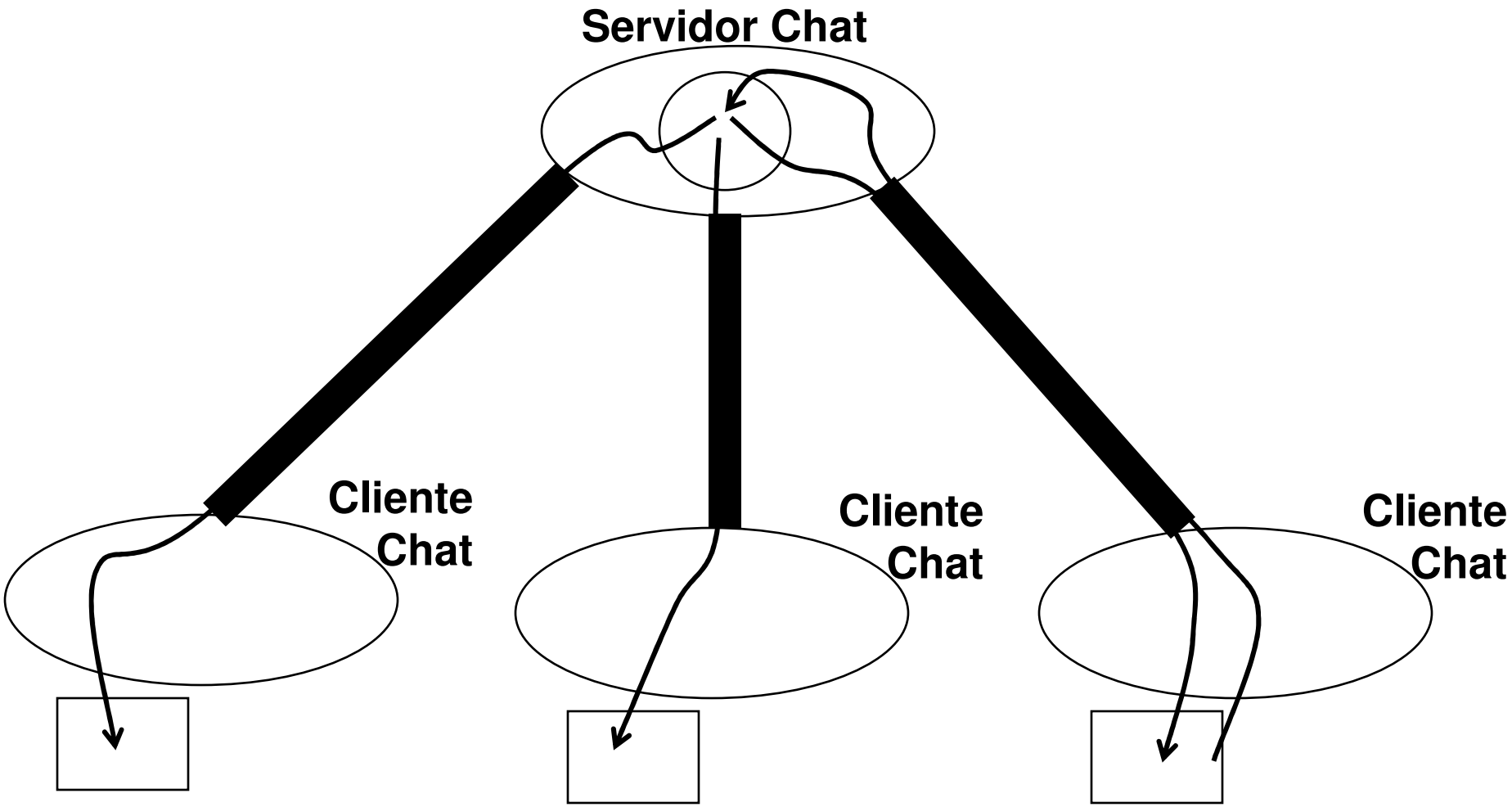
Programação Sockets



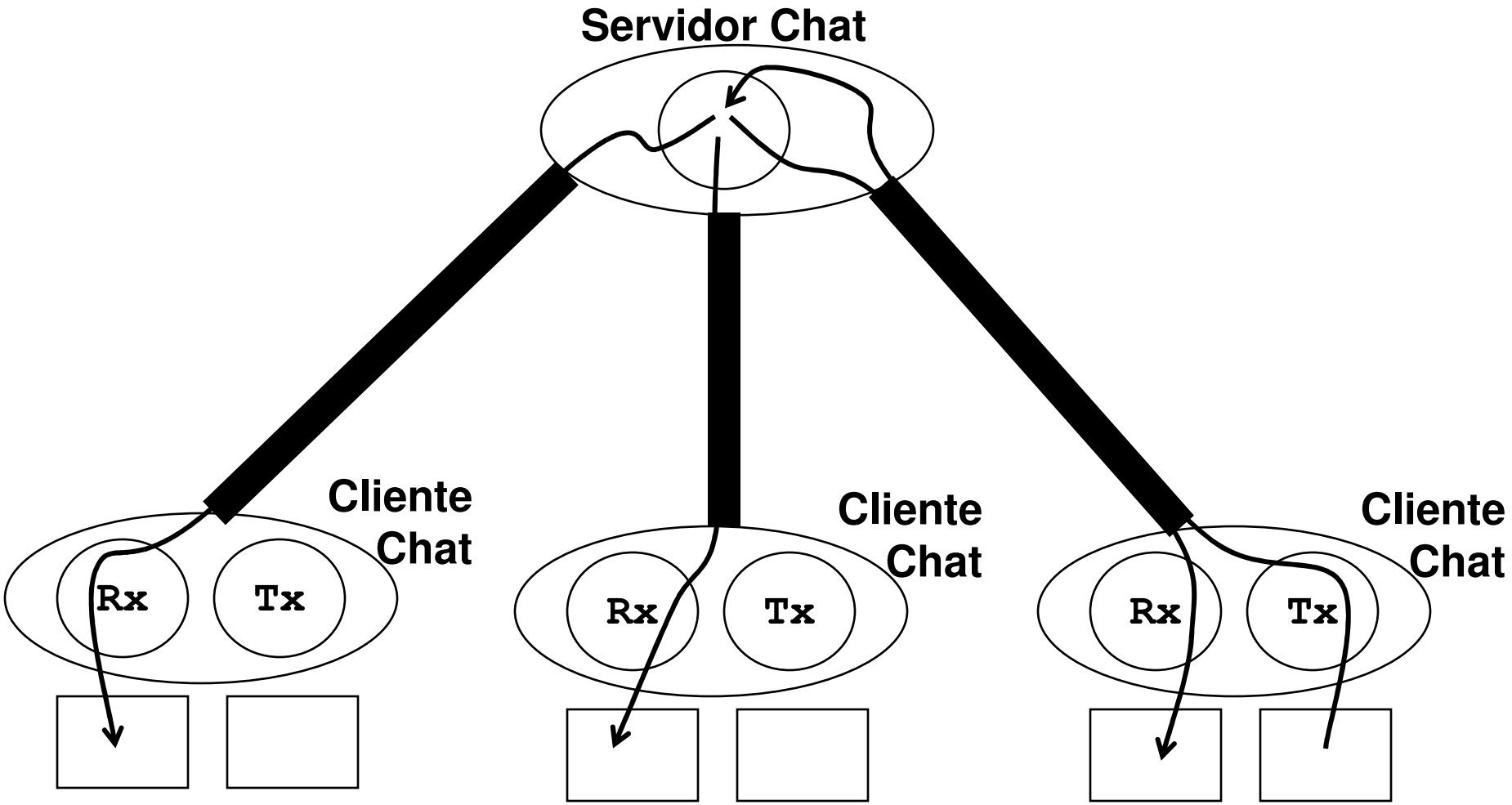
Chat UDP

- **Objetivo:**
  - Desenvolvimento de um programa chat UDP (cliente e servidor)
- **Grupo**
  - Cada grupo deve escolher implementar o servidor ou o cliente
  - Cada grupo de 2 pessoas
- **Formato do trabalho**
  - Papel A4, folhas grampeadas (não encadernar!!)
  - Página de rosto informando:
    - Nome da disciplina, título do trabalho e nome dos autores
- **Entrega:**
  - Data entrega: **23 de maio**
  - Entrega do trabalho escrito durante a aula
  - Execução do programa durante a aula
  - Serão descontados 2 pontos da nota para cada dia de aula em atraso

Chat UDP



Chat UDP


Servidor Chat
Cliente Chat
Rx
Tx
Cliente Chat
Rx
Tx
Cliente Chat
Rx
Tx

Chat UDP

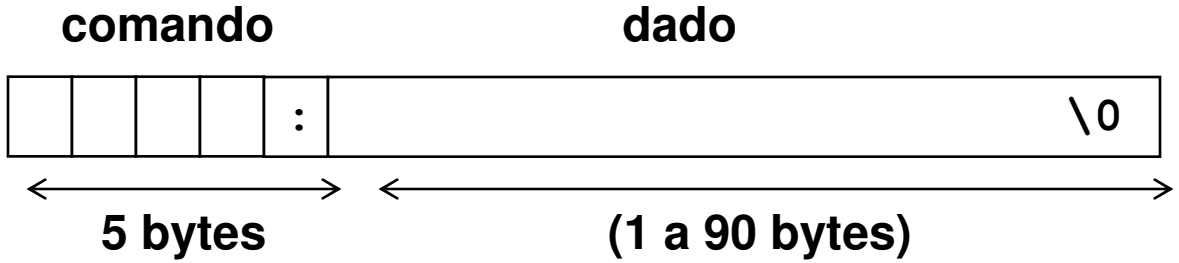
- **Servidor CHAT UDP**
  - Deve aguardar requisições na porta 10.000
  - Deve permitir sessões de chat com até 3 usuários (3 clientes chat) simultaneamente
  - Deve guardar o “socketaddress” do cliente quando receber mensagem de pedido de entrada no chat (USER)
  - Deve armazenar o nome do usuário
  - Deve verificar, para cada mensagem recebida, o “socketaddress” de origem.
  - Deve, a cada 30s, encaminhar mensagem TESTE a cada usuário com a finalidade de verificar se ainda está ativo. Caso duas mensagens de teste não sejam respondidas, deve realizar a saída do usuário do Chat
  - Quando receber uma mensagem EXIT deve enviar a seguinte mensagem a todos os clientes “<user>: Saiu”
  - Deve mostrar (para efeito de debug) todas as mensagens recebidas e encaminhadas.

Chat UDP

- **Cliente CHAT UDP**
  - Deve enviar datagramas UDP para a porta 10.000 do servidor chat
  - Deve possuir dois threads:
    - Thread transmissor:
      - Obtém a mensagem do usuário e a transmite ao servidor
    - Thread receptor:
      - Aguarda mensagens do servidor e apresenta no terminal
  - A tela de recepção deve apresentar as mensagens para o usuário da seguinte forma:
    - Maria > Olá a todos
    - Ricardo> Olá Maria
    - Jose > Olá Maria, seja bem vinda.
  - Deve, a cada 30s, encaminhar mensagem TESTE ao servidor com a finalidade de verificar se a conexão está ativa ou se o servidor está ativo. Caso duas mensagens de teste não sejam respondidas, deve mostrar na tela do usuário e terminar o programa.
  - Deve apresentar a mensagem “Número de usuários excedido” quando receber a mensagem “BUSY”

Chat UDP

- **Formato geral das mensagens**
  - Codificada em ASCII
  - Possui duas partes:
    - Comando: tamanho de 5 caracteres
    - Dado: tamanho variável, de 1 a 90 bytes (incluindo caractere '\0')



Chat UDP

- **Comandos iniciados no cliente**

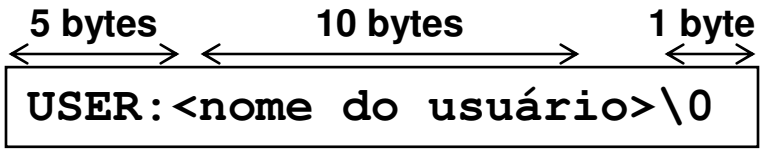
Mensagem (cliente → servidor)	Resposta (servidor → cliente)
USER (entrar no chat)	OKOK BUSY (sem slot de usuário)
UP (enviar mensagem)	(sem resposta)
EXIT	BYE (confirmação da saída)
TEST	OKOK

- **Comandos iniciados no servidor**

Mensagem (servidor → cliente)	Resposta (cliente → servidor)
DOWN (mostrar mensagem)	(sem resposta)
TEST	OKOK

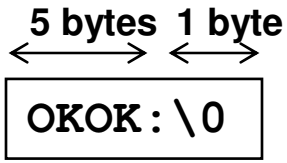
Chat UDP

- **Mensagem USER**
  - Solicitação de entrada de usuário ao chat
  - Mensagem encaminhada pelo cliente
  - Servidor chat deve armazenar os dados deste usuário:
    - Nome (até 10 caracteres)
    - Socket address
  - Servidor deve responder:
    - OKOK – Sucesso
    - BUSY – Número de usuários excedido
  - Formato:

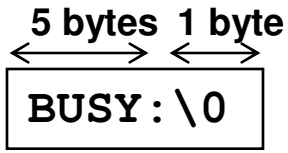


Chat UDP

- **Mensagem OKOK**
  - Confirmação de sucesso
  - Mensagem encaminhada pelo cliente ou pelo servidor

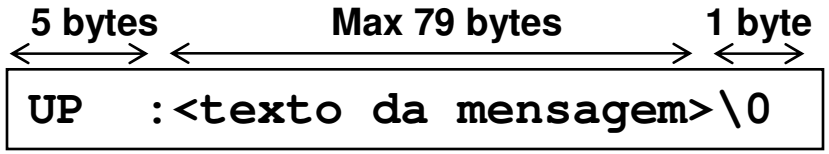


- **Mensagem BUSY**
  - Indicação de excesso de usuários
  - Mensagem encaminhada pelo servidor em resposta a USER quando há excesso de usuário e não existe slot disponível

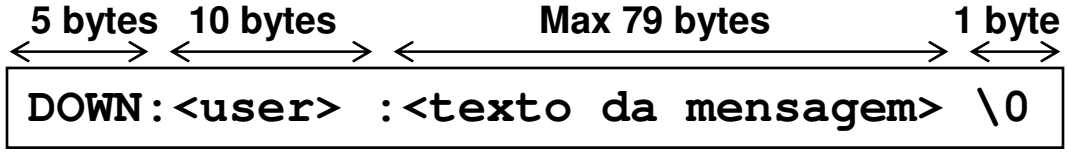


Chat UDP

- **Mensagem UP**
  - Envio de texto de mensagem do cliente ao servidor.
  - O servidor deve obter o nome do usuário de sua tabela de controle a partir do endereço socket da mensagem recebida.
  - Não existe mensagem de confirmação OKOK do servidor

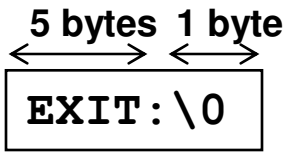


- **Mensagem DOWN**
  - Envio de texto de mensagem do servidor ao cliente.
  - Deve adicionar ao texto da mensagem o nome do usuário obtido de sua tabela de controle.
  - Não existe mensagem de confirmação OKOK do cliente

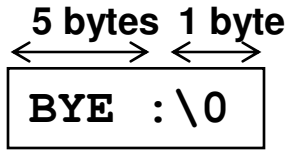


Chat UDP

- **Mensagem EXIT**
  - Pedido de saída do Chat
  - Mensagem encaminhado do cliente ao servidor
  - Servidor deve liberar slot ocupado pelo usuário
  - Servidor deve confirmar encaminhando mensagem BYE ao cliente
  - Servidor deve gerar mensagem DOWN a todos clientes “<usuário> saiu.

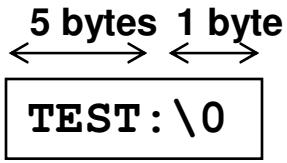


- **Mensagem BYE**
  - Confirmação de saida de cliente
  - Mensagem encaminhada do servidor ao cliente.
  - Cliente deve terminar o programa chat ao receber a mensagem BYE



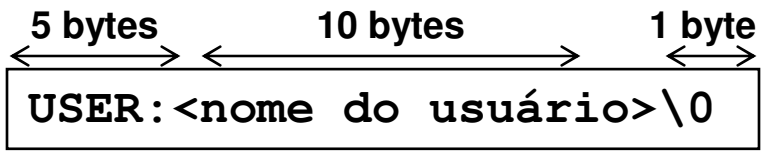
Chat UDP

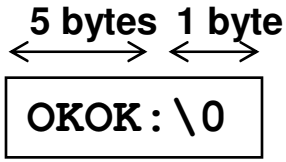
- **Mensagem TEST**
  - Pedido de teste de conecção
  - Mensagem encaminhado pelo cliente ou pelo servidor
  - Receptor da mensagem TEST deve responder com mensagem OKOK.

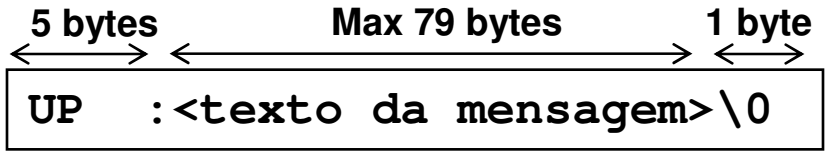


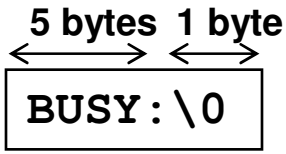
Chat UDP

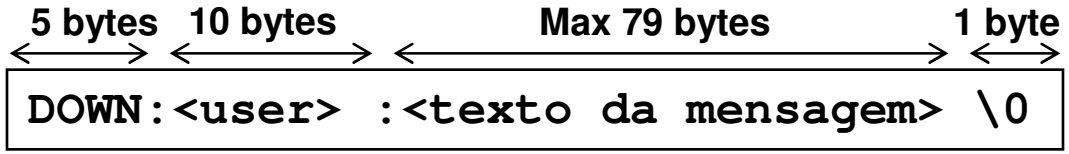
- **Resumo das mensagens**

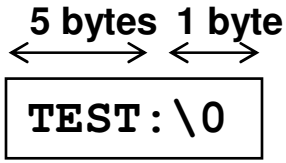


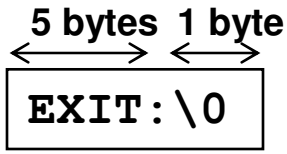


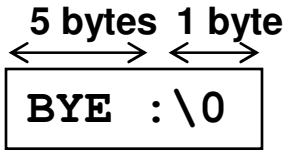




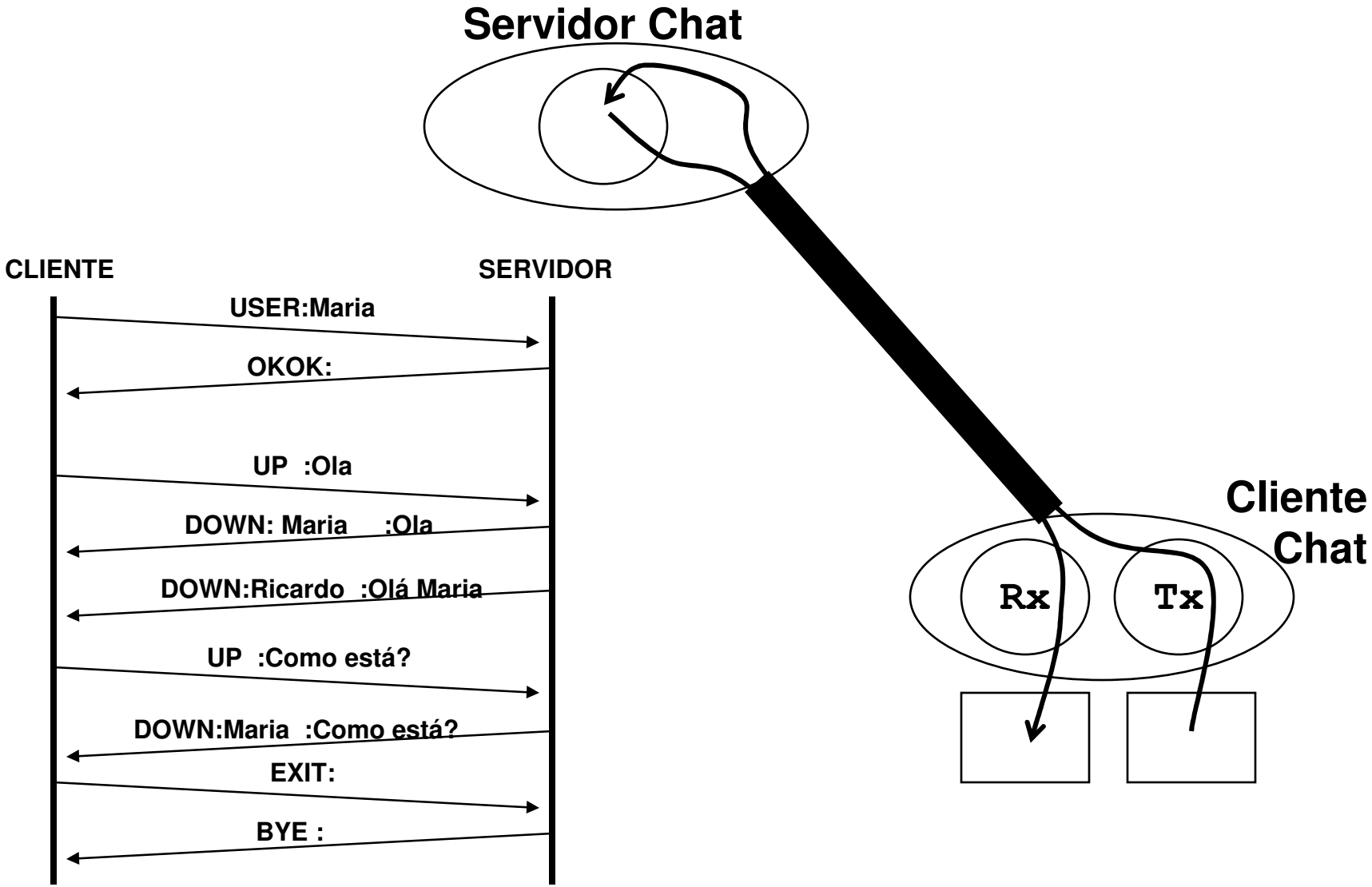








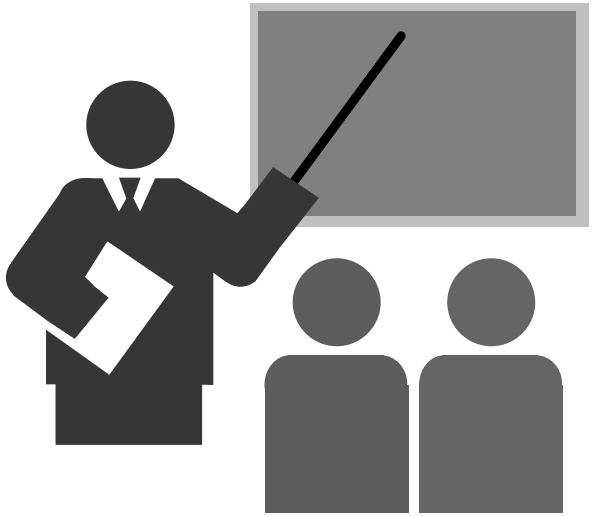
Chat UDP


Servidor Chat
CLIENTE
SERVIDOR
USER:Maria
OKOK:
UP :Ola
DOWN: Maria :Ola
DOWN:Ricardo :Olá Maria
UP :Como está?
DOWN:Maria :Como está?
EXIT:
BYE :
Cliente Chat
Rx
Tx

Chat UDP

- **Ambiente e linguagem**
  - Ambiente Linux
  - Linguagem C
  - Biblioteca pthreads
  - Interface sockets

Dicas



Dicas

- **Desenvolva seu projeto em etapas:**
  - 1ª etapa: Cliente e servidor ECHO
    - Servidor UDP: servidor ECHO
    - Cliente UDP: cliente ECHO com dois threads (transmissor e receptor)
  - 2ª etapa: Implementação do protocolo
    - Servidor UDP: atendendo somente um cliente
    - Cliente UDP: interagindo com o servidor
  - 3ª etapa: Final
    - Servidor UDP: atendendo até 3 clientes.
    - Cliente UDP: com dois threads (transmissor e receptor)

Dicas

- **Manter tabelas de controle no servidor:**
  - Manter uma tabela de controle com uma entrada para cada cliente.
  - O tamanho da tabela é o tamanho máximo de clientes
  - Esta tabela deve possuir, no mínimo, as seguintes entradas:
    - Estado da entrada da tabela (livre ou ocupada)
    - “nome do usuário” (até 10 caracteres + 1 (“\0”)
    - “socket address” do cliente

Dicas

- **Janelas**
  - Em um chat, conforme são digitadas as mensagens, são também recebidas outras mensagens, de forma concorrente.
  - Nesta situação, caso seja utilizada somente uma janela para apresentação das mensagens transmitidas (digitadas) e das mensagens recebidas, tais mensagens poderão ficar intercaladas, tornando muito confuso para o usuário.
  - Assim, devem ser utilizadas duas janelas, uma para digitar as mensagens a serem enviadas e uma outra na qual são apresentadas as mensagens recebidas dos usuários.

Dicas

- **Dicas para utilização de duas janelas:**
  - Comando para identificação do terminal corrente: “tty”
  - Trecho de código para enviar mensagens de texto para outro terminal:

```
char   terminalname[80];
FILE * terminal;

...
printf(“Entre com o nome do terminal auxliar ao chat: ”);
scanf(“¨s”,terminalname);
terminal = fopen(terminalname,“a+”);
if (terminal == NULL)
    {
    perror(“Abertura do terminal”);
    exit(1);
    }
....
fprintf(terminal, “teste de terminal \n”);
....
```

Dicas

- **Funções para desenvolvimento**
  - Utilizar fgets() ao invés de scanf()
    - #include <stdio.h>
    - char *fgets (char *string, int size, FILE *stream);
    - Evita problemas de overflow do buffer, pois gets() permite definir o tamanho do buffer.
    - A função fgets() lê caracteres até encontrar newline ou chegar ao tamanho do buffer. O newline é acrescentado à string. O caracter ´\0´ é acrescentado ao final.
  - Exemplo:

```
#include <stdio.h>
char   buffer[80];
fgets(buffer,80,stdin);
buffer[strlen(buffer)-1]=´\0´;  // retira \n
```