
Trabalho:

Servidor Chat UDP

Programação Sockets



Chat UDP

❑ **Objetivo:**

- ❖ Desenvolvimento de um programa SERVIDOR chat UDP
- ❖ O programa cliente será disponibilizado

❑ **Grupo**

- ❖ Cada grupo de 2 pessoas

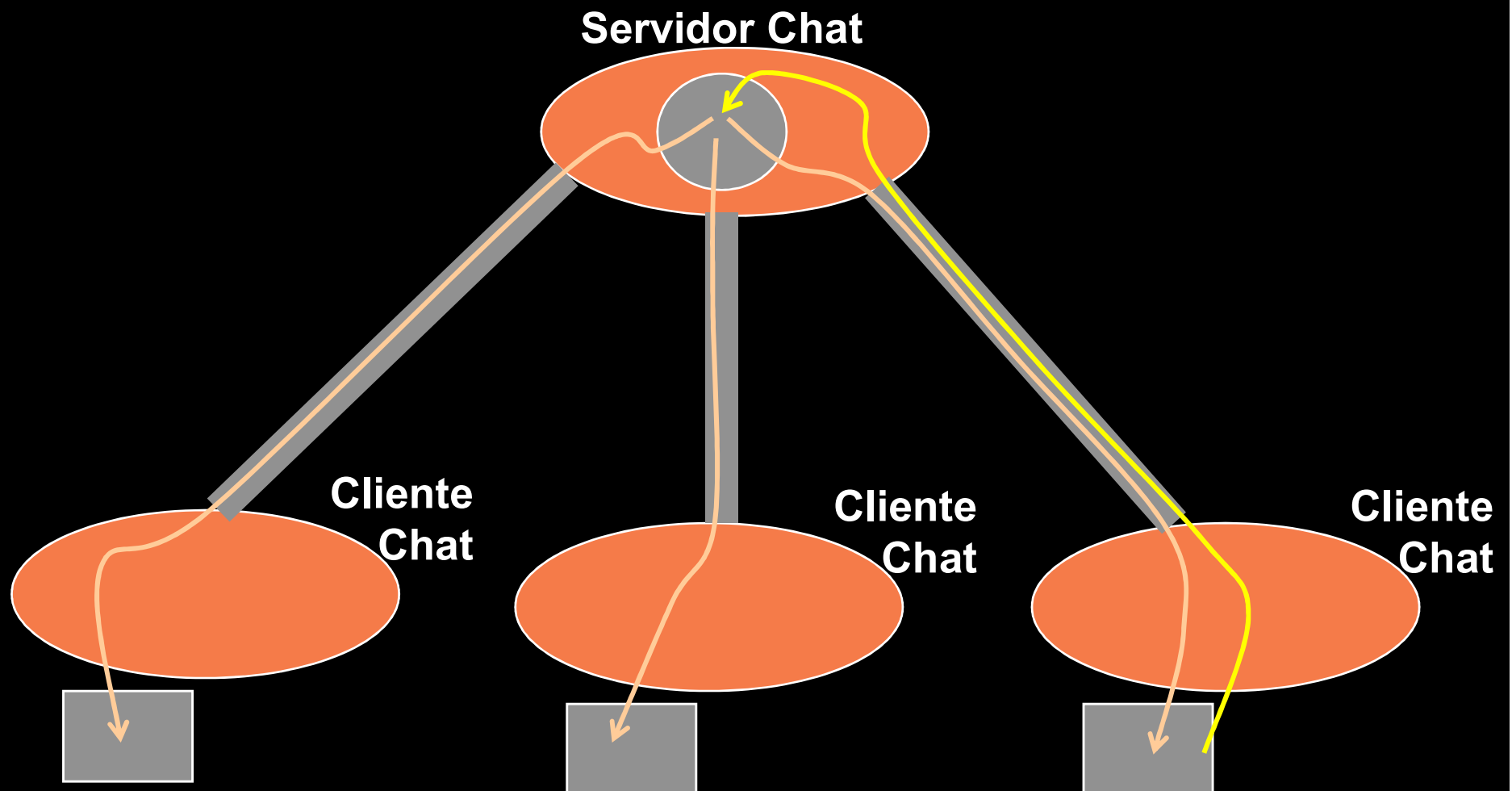
❑ **Formato do trabalho**

- ❖ Formato eletrônico, depositado no moodle
- ❖ Página de rosto informando:
 - Nome da disciplina, título do trabalho e nome dos autores

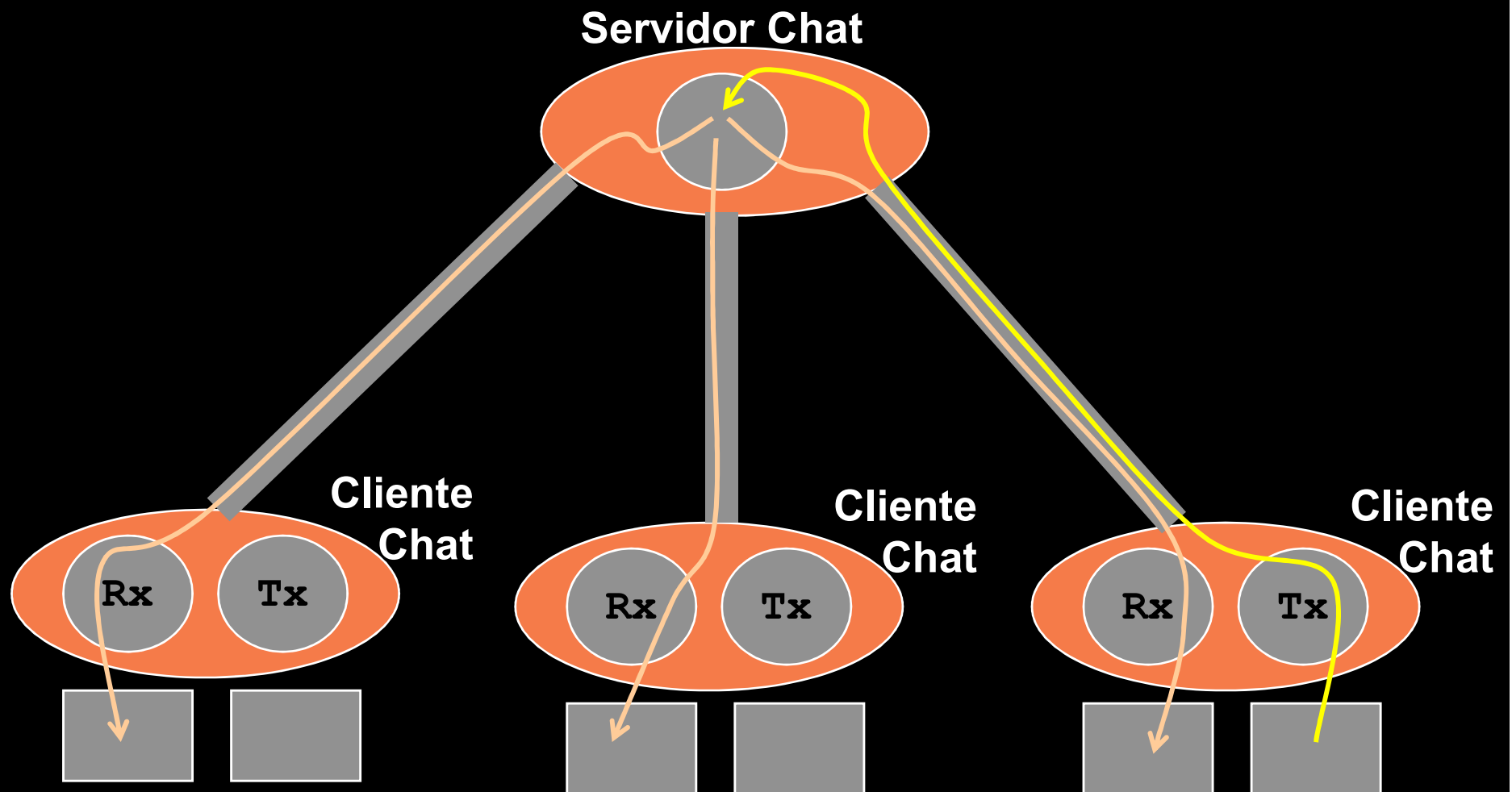
❑ **Entrega:**

- ❖ Data entrega: **5 de junho**
- ❖ Execução do programa durante a aula
- ❖ Serão descontados 2 pontos da nota para cada dia de aula em atraso

Chat UDP



Chat UDP



Chat UDP

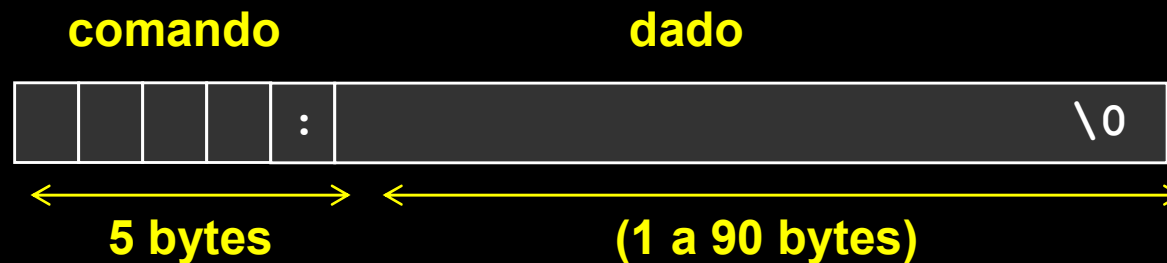
❑ Servidor CHAT UDP

- ❖ Deve aguardar requisições na porta 10.000
- ❖ Deve permitir sessões de chat com até 3 usuários (3 clientes chat) simultaneamente
- ❖ Ao receber o comando USER (conexão de usuário), deve armazenar o nome do usuário e seu endereço (socket address).
- ❖ Deve, a cada 30s, encaminhar mensagem TEST a cada cliente com usuário ativo com a finalidade de verificar se ainda está ativo. Caso duas mensagens de teste não sejam respondidas, deve realizar a saída deste usuário do Chat, enviando mensagem de saída a todos os usuários (ver a seguir)
- ❖ Quando receber uma mensagem EXIT deve enviar a seguinte mensagem a todos os usuários "<user>: Saiu"
- ❖ Deve mostrar no console (para efeito de debug) todas as mensagens recebidas e encaminhadas.
- ❖ Dicas
 - Deve verificar, para cada mensagem recebida, o "socketaddress" de origem.

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:


USER:<nome do usuário>\0

Chat UDP

❑ Mensagem OKOK

- ❖ Confirmação de sucesso
- ❖ Mensagem encaminhada pelo cliente ou pelo servidor

5 bytes 1 byte
↔ ↔
OKOK : \0

❑ Mensagem BUSY

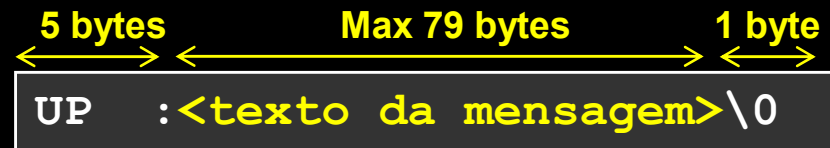
- ❖ Indicação de excesso de usuários
- ❖ Mensagem encaminhada pelo servidor em resposta a USER quando há excesso de usuário e não existe slot disponível

5 bytes 1 byte
↔ ↔
BUSY : \0

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."

5 bytes 1 byte
EXIT : \0

❑ Mensagem BYE

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

5 bytes 1 byte
BYE : \0

Chat UDP

❑ Mensagem TEST

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

5 bytes 1 byte
↔ ↔
TEST : \0

Chat UDP

□ Resumo das mensagens

5 bytes 10 bytes 1 byte
 USER:<nome do usuário>\0

5 bytes 1 byte
 OKOK:\0

5 bytes Max 79 bytes 1 byte
 UP :<texto da mensagem>\0

5 bytes 1 byte
 BUSY:\0

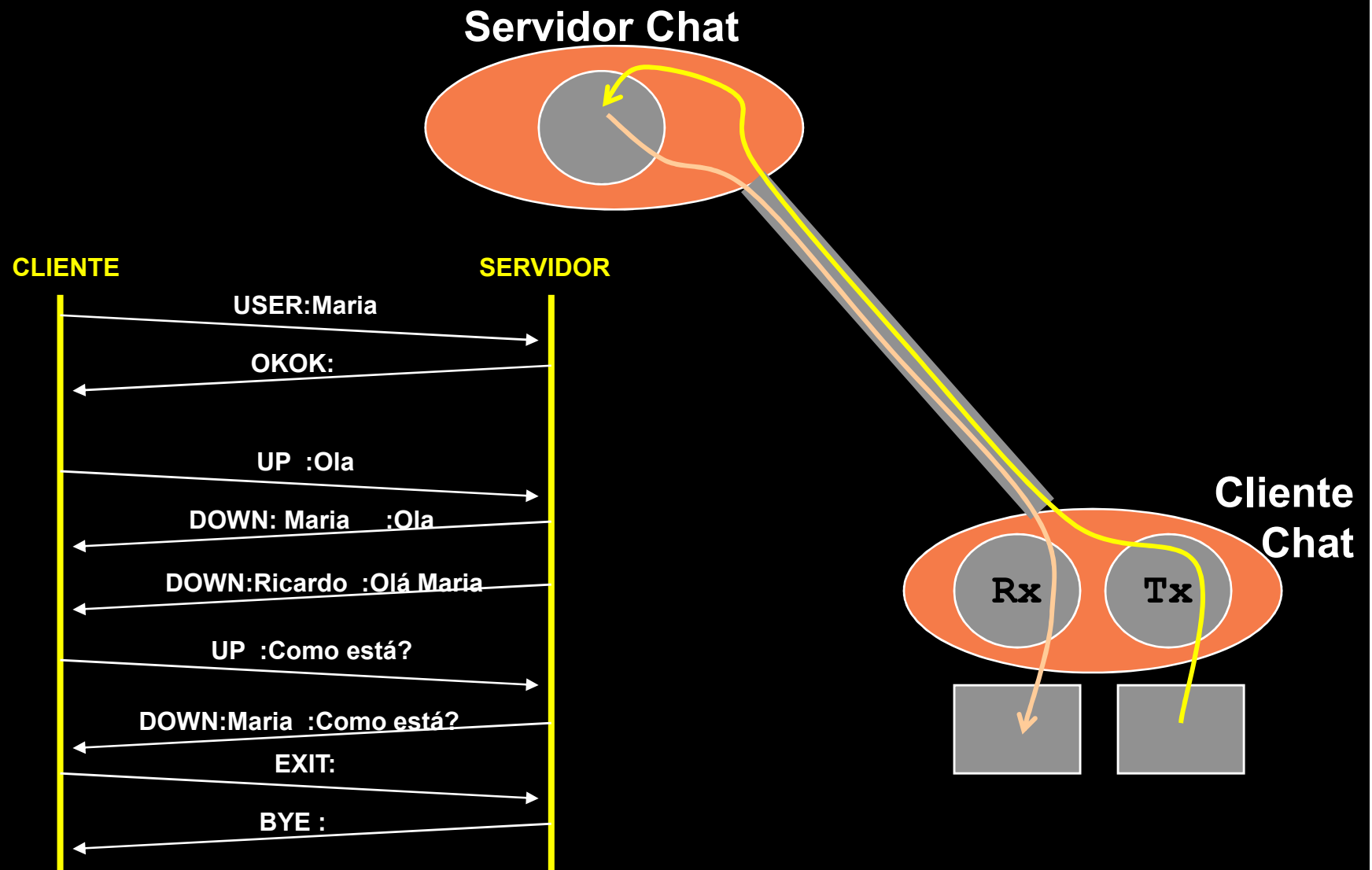
5 bytes 10 bytes Max 79 bytes 1 byte
 DOWN:<user> :<texto da mensagem> \0

5 bytes 1 byte
 TEST:\0

5 bytes 1 byte
 EXIT:\0

5 bytes 1 byte
 BYE : \0

Chat UDP

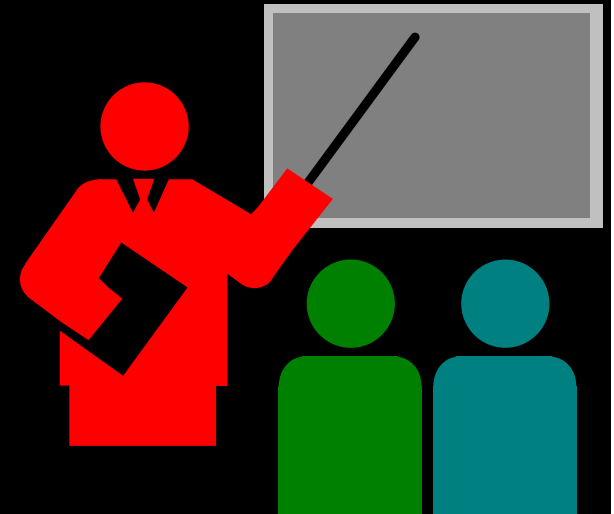


Chat UDP

❑ Ambiente e linguagem

- ❖ Ambiente Linux
- ❖ Linguagem C
- ❖ Biblioteca pthreads
- ❖ Interface sockets

Dicas



Dicas

- ❑ **Servidor UDP - Tabela de controle dos clientes:**
 - ❖ 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

❑ 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'; // sobrepõe \n
```