# Projeto de uma aplicação de chat distribuído

**Sistemas Distribuídos**
**Prof. Silvana Rossetto**

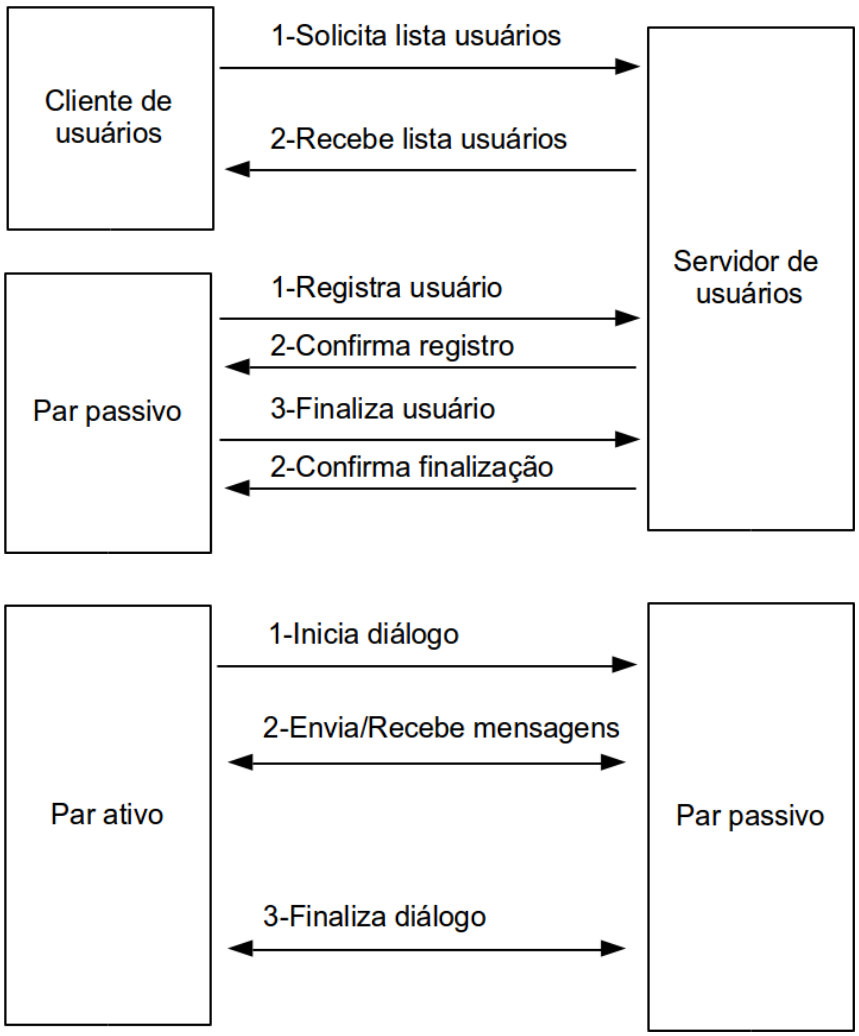
[1]DCC/IM/UFRJ — 18 de abril de 2016

## 1. Objetivos

Projetar e implementar um chat distribuído.

## 2. Arquitetura de sistema

- **Servidor de usuários**: mantem lista de usuários ativos
- **Cliente de usuários**: consulta servidor de usuários e exibe a lista de usuários ativos
- **Par passivo**: usuário que espera conexões de outros usuários e mantém diálogo
- **Par ativo**: usuário que inicia o diálogo com outro usuário



## 3. Servidor de usuários (servidor multithreading)

```
início
   1- Cria um nó servidor: <máquina local> //na porta padrão 2016
   2- Aguarda conexões
   3- Cria uma nova thread para cada conexão recebida: <socket da conexão>
      3.1- Em cada thread, recebe e trata três tipos de requisições e finaliza:
        a) Registrar usuário (identificador 1): <nome do usuário>
                                                  //string com até 20 caracteres
            //pegar o IP do usuário usando a função "getpeername" (ver exemplo abaixo)
           a.1) Envia mensagem de confirmacao da operacao: 1 (ok), 0 (erro)
        b) Enviar lista de usuários ativos (identificador 2):
             b.1) primeira mensagem com o número de usuários
             b.2) uma mensagem para cada usuário com duas strings separadas por
                  espaço em branco, a primeira com o IP e a segunda com o nome
                  do usuário (terminar com \n)
        c) Finalizar usuário (identificador 3): <nome do usuário>
           c.1) Envia mensagem de confirmacao da operacao: 1 (ok), 0 (erro)
   4- Finaliza servidor quando o administrador determinar (digitar "FIM")
fim
```

## 4. Cliente de usuários (cliente)

```
início
   1- Conecta-se com o servidor de usuários: <IP do servidor> //na porta padrão 2016
   2- Solicita lista de usuários ativos (identificador 2)
   3- Recebe lista de usuários ativos
   4- Exibe lista de usuários ativos
   5- Aguarda comando do administrador:
      5.1) Atualiza lista de usuários (volta ao passo 1)
      5.2) Finaliza aplicação
fim
```

## 5. Par passivo (servidor iterativo)

```
início
   1- Cria um nó servidor: <máquina local> //na porta padrão 2017
   2- Conecta-se com o servidor de usuários: <IP do servidor> //na porta padrão 2016
   3- Registra seu usuário (identificador 1): <nome do usuário>
                                                  //string com até 20 caracteres
   4- Aguarda conexão
   5- Interage com o "par ativo" até diálogo terminar: <FIM> (usar "select")
   6- Volta ao passo 4 ou finaliza servidor:
      6.1- Finaliza usuário no servidor de usuários (identificador 3)
      6.2- Finaliza aplicação
fim
```

## 6. Par ativo (cliente)

```
início
   1- Conecta-se com um "par passivo": <IP do par passivo> //na porta padrão 2017
   2- Envia mensagem de saudação
   3- Interage com o "par passivo" até diálogo terminar: <FIM> (usar "select")
   4- Finaliza aplicação
fim
```

Atenção: modularizar todos os códigos definindo funções principais e chamando essas funções na *main*.

## Exemplo de uso da função “getpeername”

```
...
int retcode;
unsigned int cliLen;

struct sockaddr_in cliAddr; //estrutura para guardar informações do cliente
memset((void *) &cliAddr, 0, sizeof(cliAddr)); //zera a estrutura de dados
cliLen = sizeof(cliAddr); //tamanho atual da estrutura de dados

...
for (;;) {
  cliSock = AcceptConnection(srvSock);
  //pega as informacoes do cliente que acabou de se conectar
  retcode = getpeername(cliSock, (struct sockaddr *) &cliAddr, &cliLen);
  printf("IP do cliente: %s\n", inet_ntoa(cliAddr.sin_addr));
...
```