# Sistemas Distribuídos
# Modelo Cliente-Servidor

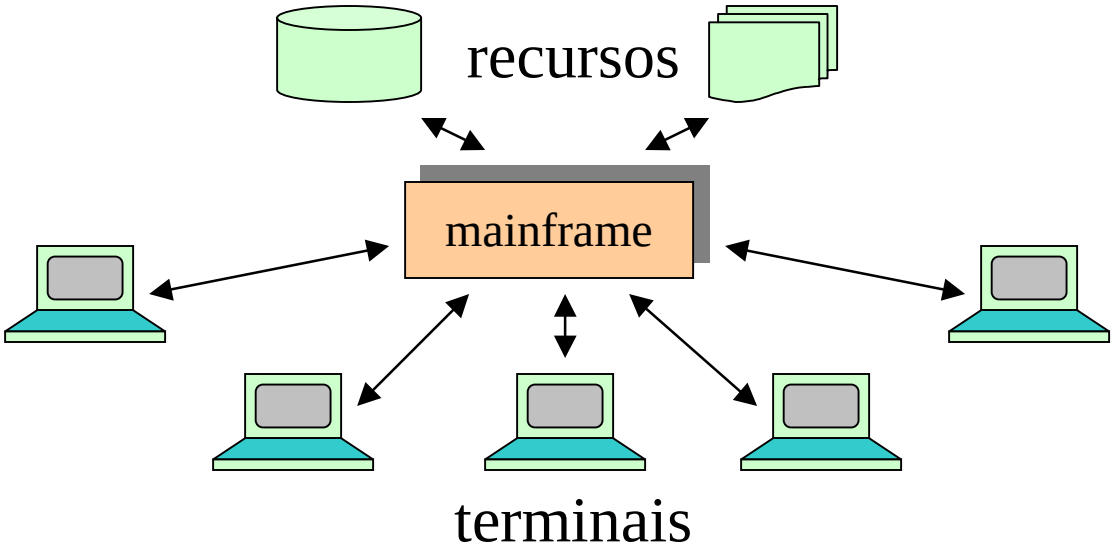
**Disciplina:** Sistemas Distribuídos
**Prof.:** Edmar Roberto Santana de Rezende

**Faculdade de Engenharia de Computação**
**Centro de Ciências Exatas, Ambientais e de Tecnologias**
**Pontifícia Universidade Católica de Campinas**

# Modelo Cliente-Servidor

## Sistema Centralizado

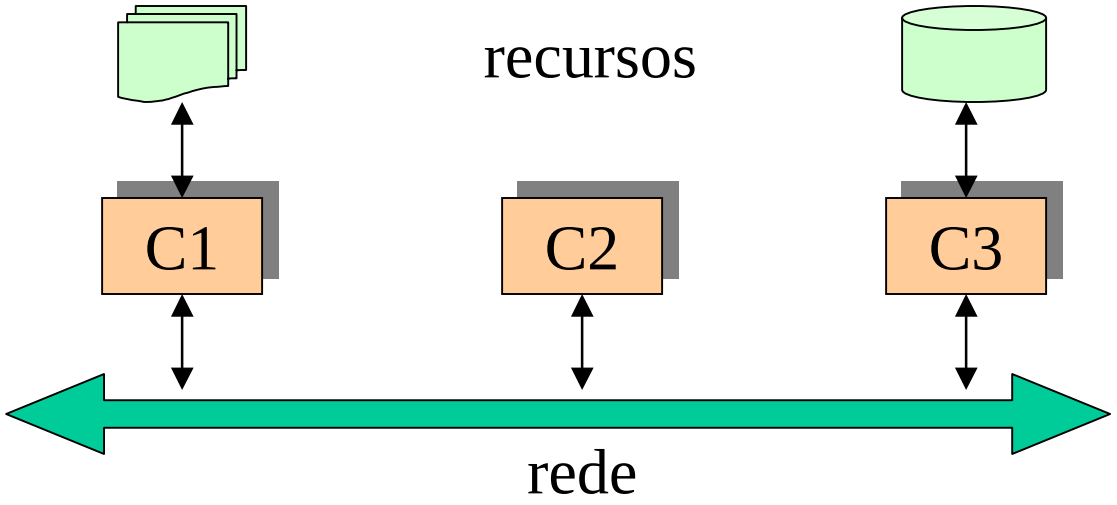
- Computador central (*mainframe*)
  + conjunto de terminais
  + recursos centralizados



# Modelo Cliente-Servidor

Sistema Distribuído

- Grupo de computadores
  + suporte de comunicação
  + recursos compartilhados



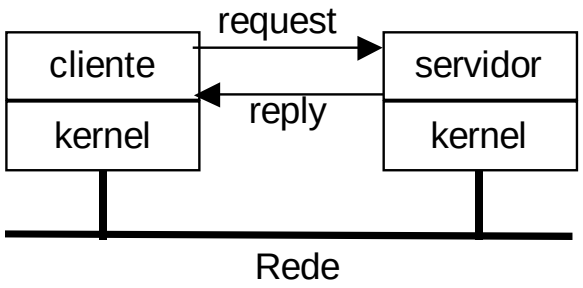
# Modelo Cliente-Servidor

## Definição

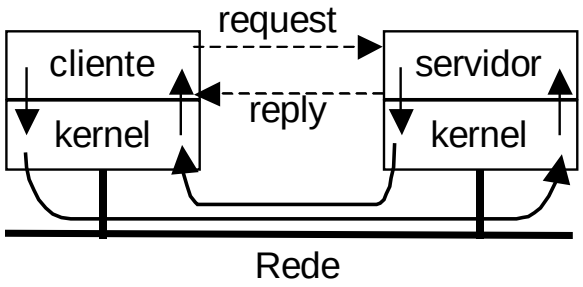
- Processamento cooperativo de requisições submetidas por um *cliente* a um *servidor* que as processa e retorna um resultado
  - *clientes*: processos que requisitam serviços;
  - *servidores*: processos que recebem requisições, realizam uma operação e retornam serviços
- Uma máquina pode executar:
  - um simples processo (cliente ou servidor)
  - múltiplos clientes
  - múltiplos servidores
  - uma mistura dos dois (múltiplos clientes e servidores)

# Modelo Cliente-Servidor

## Definição



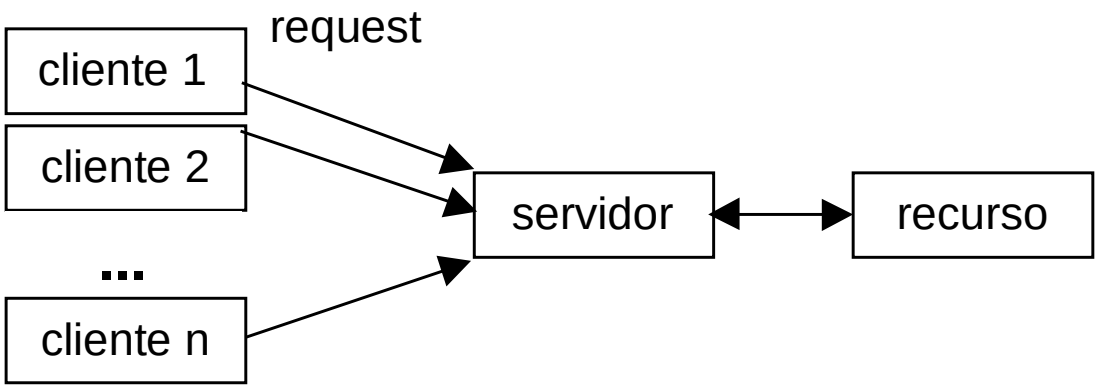

Visão lógica da comunicação cliente/servidor




Troca de mensagens que possibilita a comunicação cliente/servidor.

# Modelo Cliente-Servidor

## Definição




Vários clientes acessando um recurso de forma transparente através do servidor

# Modelo Cliente-Servidor

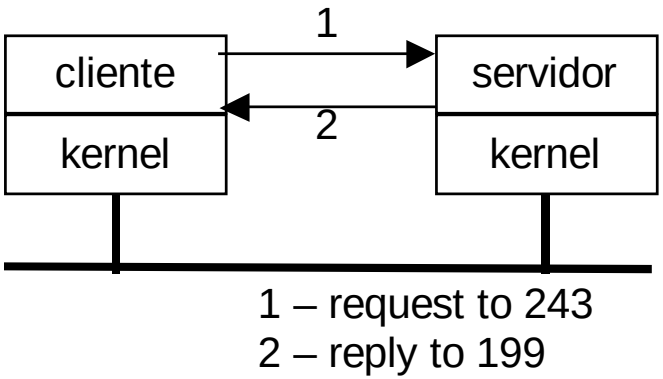
## Endereçamento

- Para que um cliente possa mandar uma mensagem a um servidor ele precisa saber seu endereço
- Existem várias formas de endereçamento
- Principais:
  - por número de máquina
  - por processo
  - por nomes ASCII obtidos de um servidor de nomes (*name server*)

# Modelo Cliente-Servidor

## Endereçamento

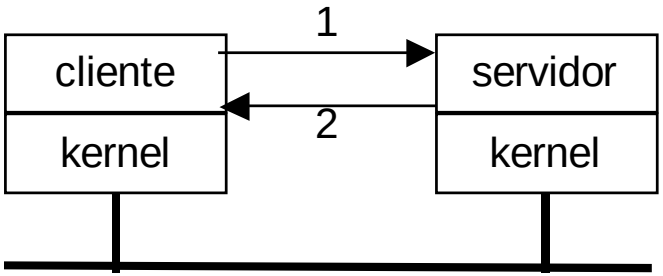
- Endereçamento por máquina
  - Utiliza o número da máquina para endereçamento
    - ☹ um processo por máquina
    - ☹ não é transparente
    - ☺ simplicidade



# Modelo Cliente-Servidor

## Endereçamento

- Endereçamento por máquina
  - Pode-se utilizar uma combinação entre número da máquina e número do processo
    - ☹ não é transparente
    - ☺ simplicidade
    - ☺ não precisa de coordenação global (não há ambiguidade entre processos)




1 – request to 243.0
2 – reply to 199.4

# Modelo Cliente-Servidor

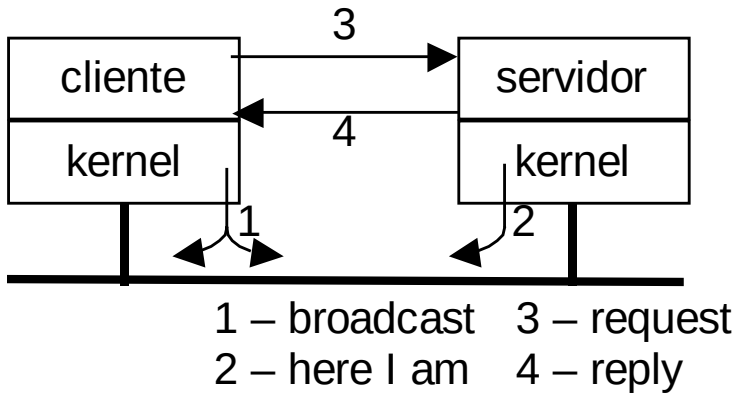
## Endereçamento

- Endereçamento por processo
  - Associar a cada processo um endereço único que não contém o número da máquina
  - Duas alternativas para a escolha do número do processo:
    - Processo centralizado responsável pela alocação de endereços
      - ☺ transparência
      - ☹ escalabilidade
    - Cada processo pega seu próprio endereço aleatoriamente de um grande espaço de dados
      - ☺ transparência
      - ☺ escalabilidade

# Modelo Cliente-Servidor

## Endereçamento

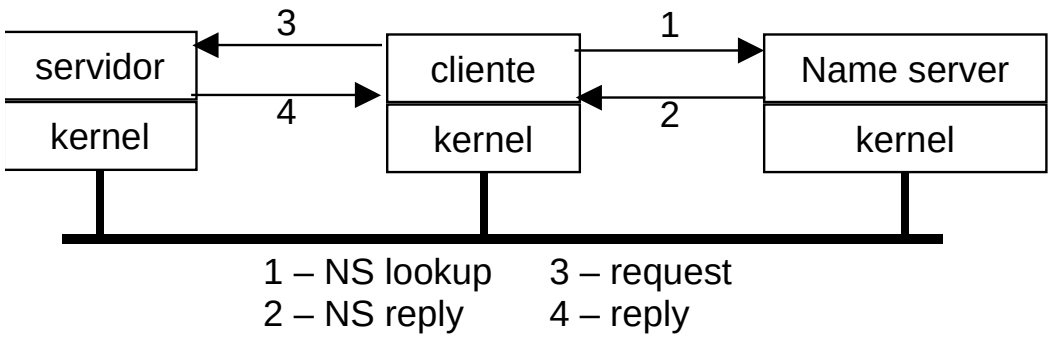
- Endereçamento por processo
  - O cliente faz *broadcast* de um pacote especial para localização
  - O kernel que contém o processo devolve uma mensagem do tipo *here I am*
    - ☺ transparência
    - ☹ gera carga extra no sistema



# Modelo Cliente-Servidor

## Endereçamento

- Endereçamento por *name server*
  - Utiliza uma máquina extra para mapear nomes de serviços em endereços de máquinas
  - servidores são referenciados como *strings* e estas são embutidas nos programas
    - ☺ transparência
    - ☹ requer um componete centralizado



# Modelo Cliente-Servidor

## Primitivas Bloqueantes e Não-bloqueantes

- Primitivas Bloqueantes (síncronas)
  - a instrução seguinte ao *send* não é executada até que a mensagem tenha sido completamente enviada
  - a chamada *receive* não retorna o controle até que a mensagem tenha sido recebida e colocada no *buffer*

- Primitivas Não-bloqueantes (assíncronas)
  - o *send* retorna o controle imediatamente, antes da mensagem ser realmente enviada
  - o *receive* passa para o kernel o ponteiro para o *buffer* e retorna imediatamente, antes de receber a mensagem

# Modelo Cliente-Servidor

## Primitivas Bloqueantes e Não-bloqueantes

- Primitivas Bloqueantes (síncronas)
  - processo fica bloqueado durante a transferência da mensagem
  - melhor opção para envio de mensagens em condições normais:
    - ☺ simples de entender
    - ☺ simples de implementar
    - ☺ performance para envio de mensagem
    - ☹ CPU fica ociosa durante a transmissão

# Modelo Cliente-Servidor

## Primitivas Bloqueantes e Não-bloqueantes

- Primitivas Não-bloqueantes (assíncronas)
  - Primitivas não-bloqueantes com cópia
    - o kernel copia a mensagem para um *buffer* interno e então libera o processo para continuar
      - ☹ desempenho
      - ☺ sobrepor processamento e transmissão da mensagem
  - Primitivas não-bloqueantes com interrupção
    - interrompe o processo que enviou a mensagem quando o *buffer* estiver livre para reutilização
      - ☹ programação difícil
      - ☺ desempenho

# Modelo Cliente-Servidor

## Bufferização

- Primitivas não-bufferizadas
  - O *buffer* para armazenar a mensagem deve ser especificado pelo programador
  - Existem duas estratégias a serem empregadas no caso de um *send* do cliente, sem um *receive* do servidor:
    - descartar mensagens inesperadas
    - temporariamente manter mensagens inesperadas
- Primitivas bufferizadas
  - Existe um *buffer* para armazenar mensagens inesperadas
  - A primitiva de bufferização mais empregada define estruturas de dados chamadas *mailbox*

# Modelo Cliente-Servidor

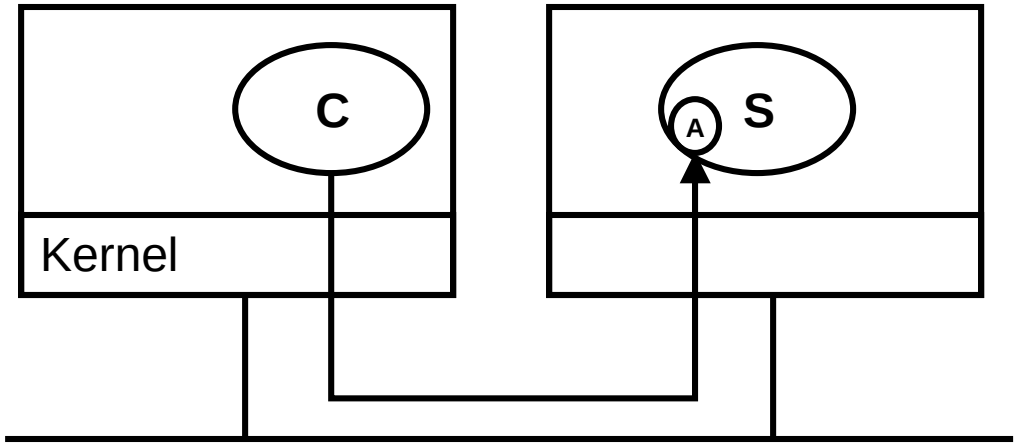
## Bufferização

- Primitivas não-bufferizadas
  - Descartando mensagens
    - ☺ Implementação
    - ☹ O cliente pode ficar tentando reenviar mensagens
    - ☹ Podem ocorrer casos em que o cliente desista de enviar a mensagem
  - Temporariamente mantendo mensagens
    - ☺ reduz a chance da mensagem ser descartada
    - ☹ introduz problema de gerenciamento e armazenamento de mensagens inesperadas

# Modelo Cliente-Servidor

## Bufferização

- Primitivas não-bufferizadas
  - Passagem de mensagem não-bufferizada



# Modelo Cliente-Servidor

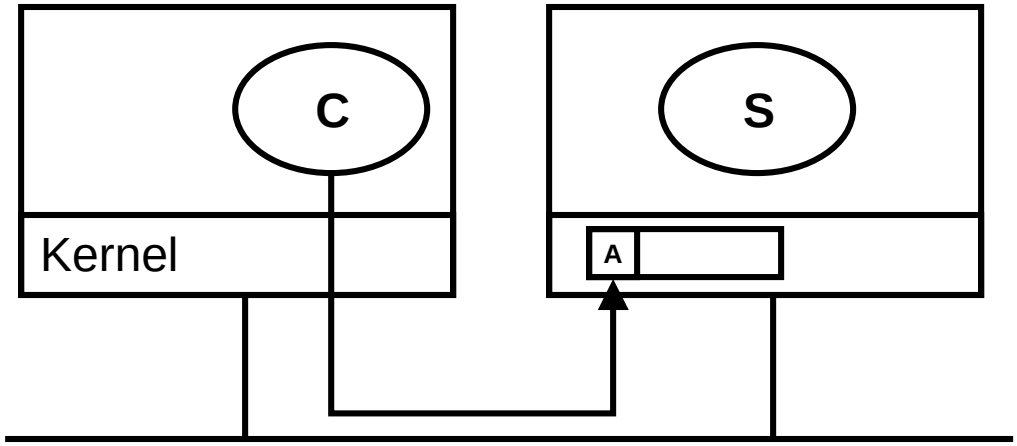
## Bufferização

- Primitivas bufferizadas
  - Um processo que está interessado em receber mensagens avisa ao kernel para criar uma *mailbox* informando o endereço de origem das mensagens
  - Todas as mensagens são colocadas na *mailbox* e uma chamada *receive* simplesmente remove mensagens dela
    - ☺ reduz ainda mais a chance de mensagens serem descartadas
    - ☹ *mailboxes* são finitas e podem necessitar de estratégias análogas às adotadas anteriormente

# Modelo Cliente-Servidor

## Bufferização

- Primitivas bufferizadas
  - Passagem de mensagem bufferizada



# Modelo Cliente-Servidor

## Confiabilidade

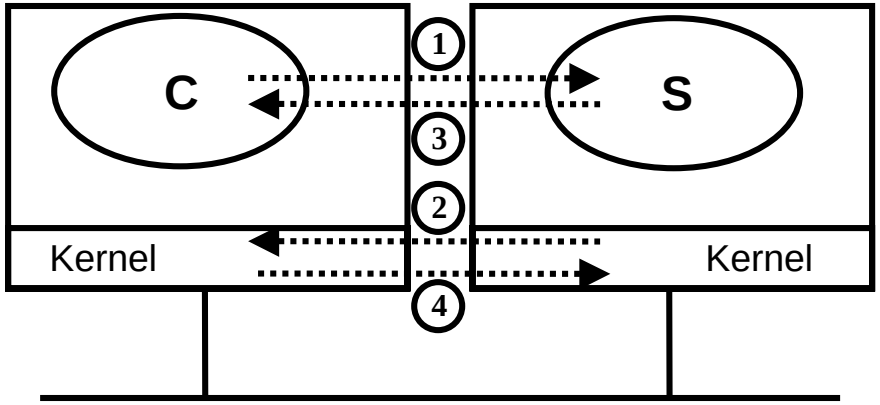
- Três diferentes alternativas podem ser utilizadas:
  - assumir que as primitivas não são confiáveis, alterando a semântica do *send*
    - o sistema não garante que as mensagens são enviadas
    - o usuário fica responsável por implementar comunicação confiável
  - primitivas confiáveis com mecanismos de *acknowledgment* do tipo:
    *Request – Ack – Reply – Ack*
  - primitivas confiáveis com mecanismos de *acknowledgment* do tipo:
    *Request – Reply – Ack*
- Combinações podem ser obtidas entre os mecanismos

# Modelo Cliente-Servidor

## Confiabilidade

- ***Request – Ack – Reply – Ack***
  - somente quando o *Ack* é recebido, o processo é liberado
  - o *Acknowledgment* é feito entre *kernels* (transparente para o cliente ou servidor)
  - um *Request/Reply* com este mecanismo necessita de quatro mensagens

1. ***Request***
2. ***Ack***
3. ***Reply***
4. ***Ack***

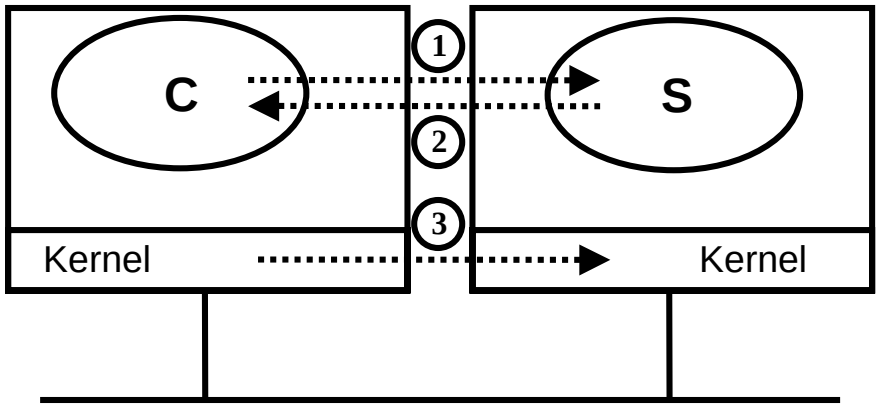


# Modelo Cliente-Servidor

## Confiabilidade

- ***Request – Reply – Ack***
  - o *Reply* serve como um *Ack*
  - o cliente fica bloqueado até a mensagem de *Reply*
  - se a mensagem de *Reply* demorar, o cliente reenvia a requisição
  - em alguns *kernels* não é necessário o *Ack*

1. ***Request***
2. ***Reply***
3. ***Ack***



# Modelo Cliente-Servidor

## Alternativas de Projeto

| | | | |
|---|---|---|---|
| Endereçamento | **por número de máquina** | **por processo** | **por nomes ASCII obtidos de um *name server*** |
| Bloqueante | **primitivas bloqueantes** | **primitivas não-bloqueantes com cópia para o *kernel*** | **primitivas não-bloqueantes com interrupção** |
| Bufferização | **não-bufferizado, descartando mensagens inesperadas** | **não-bufferizado, mantendo temporariamente mensagens inesperadas** | ***mailboxes*** |
| Confiabilidade | **não-confiáveis** | ***Request-Ack-Reply-Ack*** | ***Request-Reply-Ack*** |

# Modelo Cliente-Servidor

## Outras questões

- As redes têm um tamanho máximo de pacote
  - mensagens maiores devem ser quebradas

- O *acknowledgment* pode ser utilizado:
  - por pacote, ou
  - por mensagem

  → Dependendo da taxa de erros da rede

# Modelo Cliente-Servidor

## Exemplo de Protocolo

- Pacotes normalmente empregados no protocolo de comunicação

| | | | | |
|---|---|---|---|---|
| REQ | Request | Cliente | Servidor | O cliente quer serviço |
| REP | Reply | Servidor | Cliente | Resposta do servidor para o cliente |
| ACK | Ack | Cliente/Servidor | Outro | O pacote anterior chegou |
| AYA | Are you alive? | Cliente | Servidor | Verifica se o servidor está OK |
| IAA | I am alive | Servidor | Cliente | O servidor está OK |
| TA | Try Again | Servidor | Cliente | O servidor não tem espaço |
| AU | Address Unknown | Servidor | Cliente | Nenhum processo usa o endereço |

# Modelo Cliente-Servidor

## Exemplo de Protocolo

- Alguns exemplos de comunicação: