
Sistemas Distribuídos

Modelo Cliente Servidor
Sockets

Conceitos Iniciais

Nomenclaturas

Cliente / Servidor

- Cliente
 - Aplicação que tem por finalidade enviar requisições a um servidor (*software*) em uma máquina remota
- Servidor
 - Aplicação dedicada em receber as requisições dos clientes, processá-las e enviar as respostas

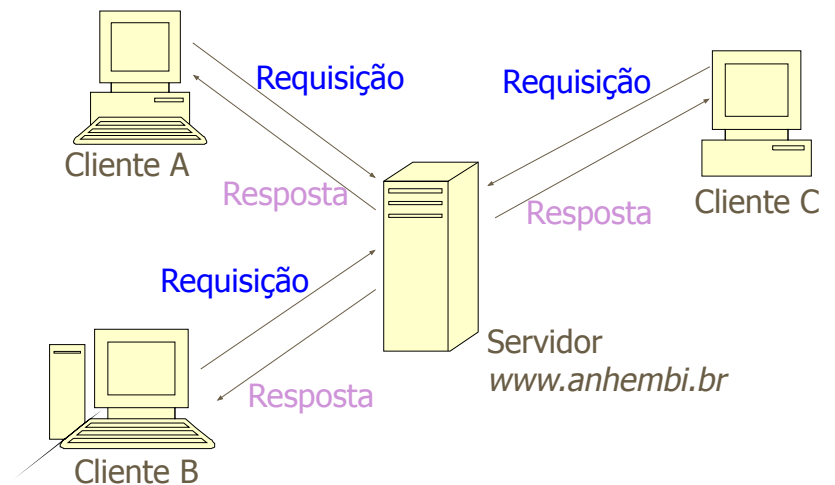
Algumas Terminologias

- Host
 - Computador ou máquina conectado à Web
- Conexão
 - Canal de comunicação entre dois hosts
- Pacote
 - Unidade básica de comunicação na rede

Algumas Terminologias

- IP
 - *Internet Protocol*, protocolo que coordena a remessa de pacotes entre os hosts
- Endereço IP
 - Endereço numérico de 32 bits (IP v4) representando um host na Internet
- DNS
 - Domain Name System – serviço responsável por traduzir nomes de hosts em endereços IP
- TCP
 - *Transmission Control Protocol* – protocolo que estabelece um canal de comunicação confiável e bidirecional

Cliente / Servidor

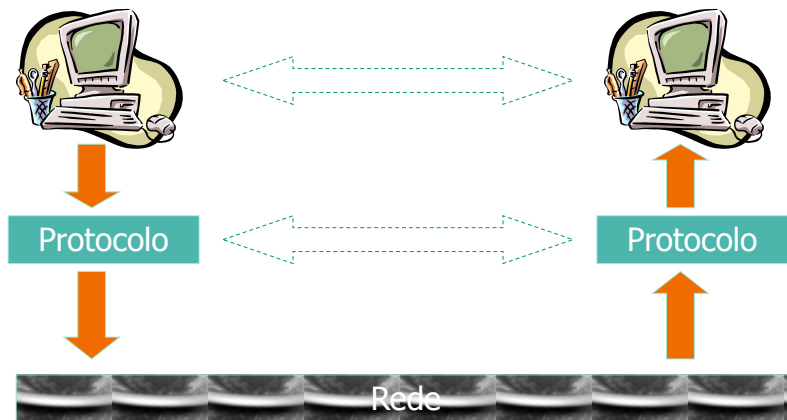


Protocolos

Introdução

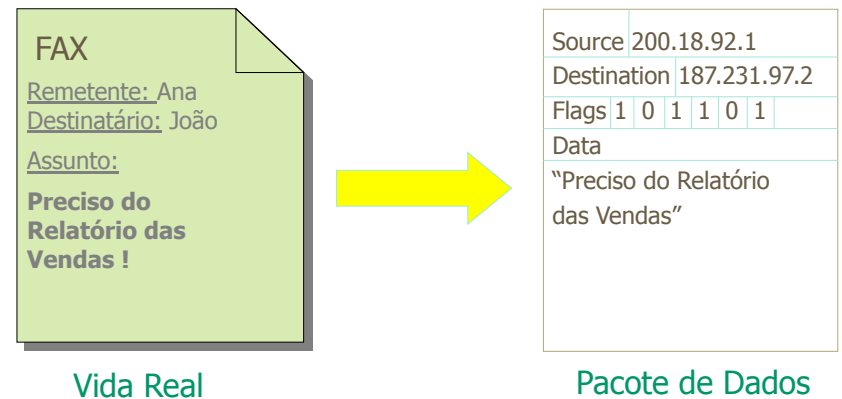
- O que é um protocolo de comunicação?
 - Conjunto de regras e procedimentos para que duas entidades distintas possam estabelecer um canal de comunicação
 - Etapas de estabelecimento e finalização de conexão
 - Cabeçalhos indicativos de numeração e ordem dos pacotes
 - Endereço do emissor e receptor

Protocolos na Internet



Prof. Emerson Paduan: emerson@paduan.dev.br

Comparação



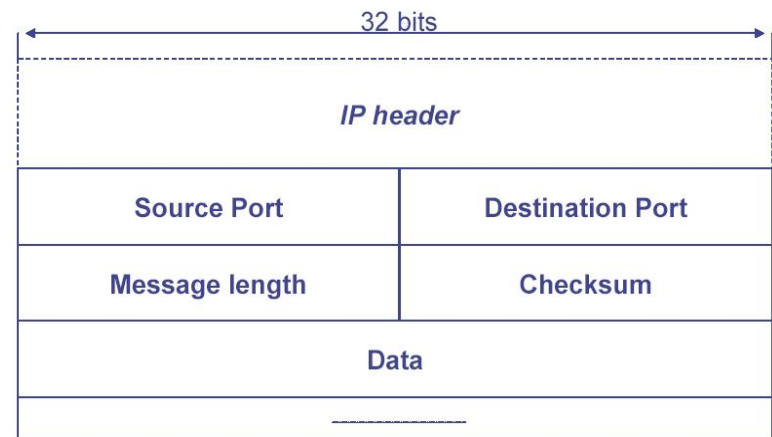
Prof. Emerson Paduan: emerson@paduan.dev.br

Protocolo UDP

- **User Datagram Protocol (UDP):**
- Protocolo não orientado à conexão;
- Não há garantia de entrega dos dados (não há mensagens de confirmação);
- Perdas durante as transmissões não são tratadas por este protocolo;
- Usado em redes com alta confiabilidade, onde as taxas de perda são baixas;

Protocolo UDP

- **Header UDP:**

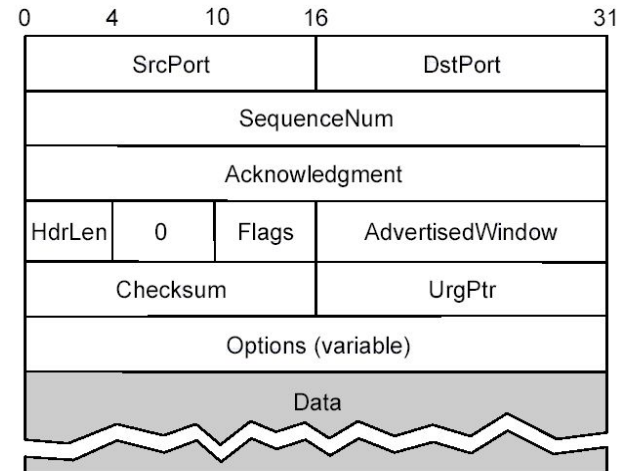


Protocolo TCP/IP

- Protocolo atualmente utilizado na Internet.
- Baseado no modelo OSI da ISO (7 camadas)
- TCP – protocolo de transporte que utiliza os serviços do protocolo IP para garantir estabelecimento de conexões e integridade de dados
- IP – protocolo de rede responsável pelo endereçamento das máquinas (endereço IP) e rotas entre dispositivos.

Protocolo TCP

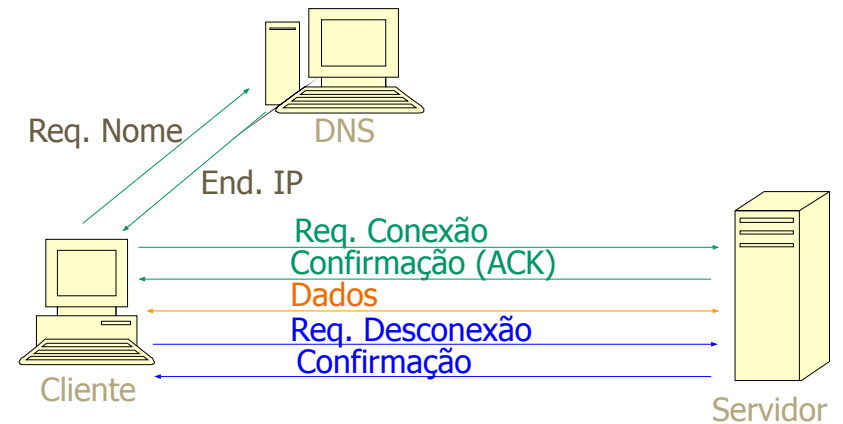
- **Header TCP:**



Protocolos TCP

- **Transmission Control Protocol (TCP):**
- Protocolo orientado à conexão:
 - Exige o estabelecimento de um canal lógico para iniciar a transmissão de dados, em 3 fases:
 - Fase de conexão
 - Fase de dados
 - Fase de desconexão
- Exemplos de aplicação:
 - TELNET, Web Browser, ...

Etapas em uma Conexão TCP



Sockets

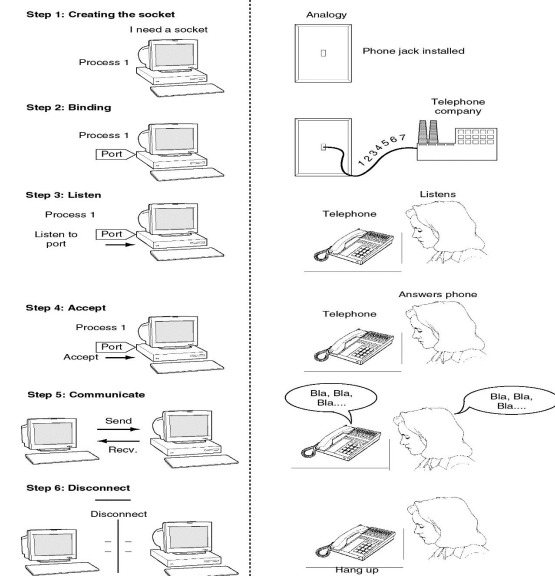
Introdução: Sockets

- Para estabelecer a Comunicação Interprocesso nos Sistemas Distribuídos e para permitir que processos se comuniquem na troca de dados ou acessos a recursos ou serviços em processadores remotos, se faz necessário o uso de um mecanismo de serviços de transporte;
- Um dos mecanismos mais utilizado é o *Socket*;
- *Sockets* é a maneira mais popular de utilizar as funcionalidades de comunicação TCP/IP;
- Todos os mecanismos *Sockets* são gerenciados pela camada de transporte;
- Existem diversas APIs *Sockets* (*Application Program Interface*) e as mais populares são do ambiente Unix, bem como a *WinSock* do Windows.

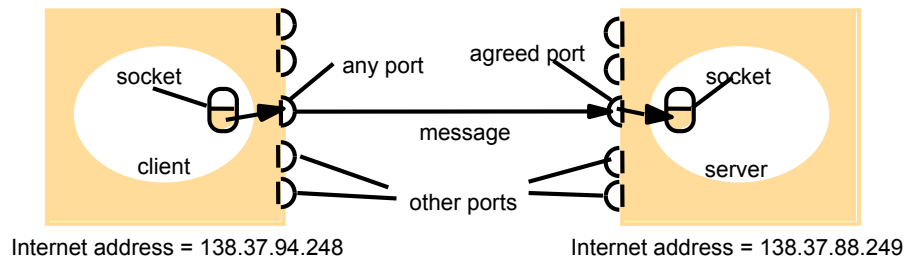
Socket: Definição

- Um *Socket* é um ponto final (*endpoint*) de um canal bidirecional de comunicação entre dois programas rodando em uma rede;
- Cada *Socket* tem os seguintes endereços de *endpoint*:
 - **Endereço local** (número da porta) que refere-se ao endereço da porta de comunicação para camada de transporte;
 - **Endereço global** (nome *host*) que refere-se ao endereço do computador (*host*) na rede.

Socket: Uma analogia



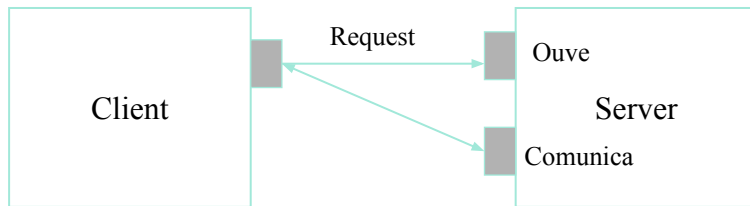
Sockets e Portas



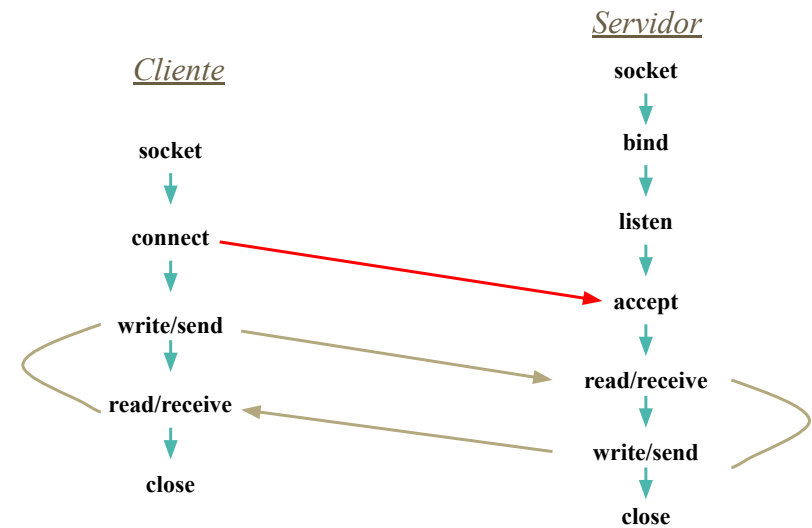
Socket: Conexão

- O servidor apenas fica “ouvindo” o *Socket* aguardando um pedido de conexão do cliente;
- O cliente sabe o nome do *host* e qual porta está associada à aplicação servidora;
- Assim que o servidor aceitar a conexão, este cria um novo *Socket* (e conseqüentemente o associa a uma nova porta) e pode ficar esperando novas conexões no *Socket* original enquanto atende às requisições do cliente pelo novo *Socket*.

Socket: Conexão



Socket: Comunicação C/S



Socket: Comunicação C/S

- **Servidor:**

- Efetua a criação de um *Socket*;
- Associa o *Socket* a um endereço local;
- Aguarda por conexões da parte cliente;
- Aceita conexões;
- Lê requisições;
- Opcionalmente envia resposta;
- Fecha o *Socket*.

Socket: Comunicação C/S

- **Cliente:**

- Efetua a criação do *Socket*;
- Estabelece a conexão;
- Envia a requisição;
- Opcionalmente aguarda resposta;
- Fecha o *Socket*.

API Sockets: Comunicação C/S

- **Socket:** (cliente e servidor)
 - Cria um *Socket* e retorna um descritor;
 - O descritor é a referência para que as outras funções utilizem o *Socket* criado.
- **Bind:** (servidor)
 - Provê o número da porta que o servidor espera contato;
 - Função utilizada apenas pelo servidor, uma vez que associa um determinado endereço IP e porta TCP ou UDP para o processo servidor.

APIs Sockets: Comunicação C/S

- **Listen:** (servidor)
 - Indica ao sistema operacional para colocar o *Socket* em modo de espera (passivo) para aguardar conexões de clientes.
- **Accept:** (servidor)
 - Cria um novo *Socket* a partir do estabelecimento de uma conexão para iniciar a comunicação (leitura e escrita).
- **Connect:** (cliente)
 - Função que o cliente utiliza para se conectar ao socket de um servidor.

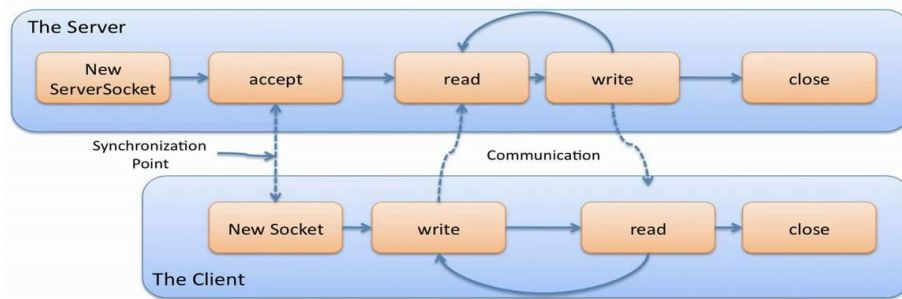
APIs Sockets: Read e Write

- **Read:**
 - Lê o conteúdo do buffer associado ao *Socket*.
- **Write:**
 - Escreve dados em um buffer associado ao *Socket*.
- **Close:** (cliente e servidor)
 - Informa ao sistema operacional para terminar o uso de um *Socket*.

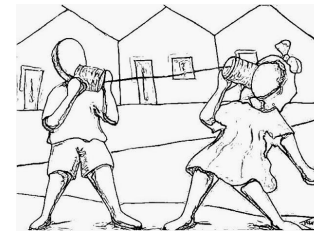
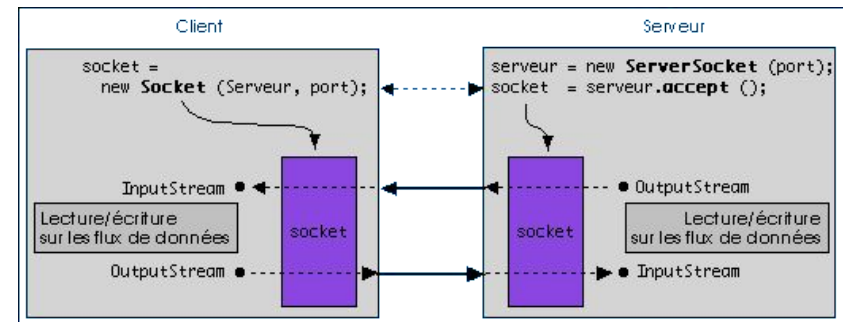


Modelo

Java Socket Overview



Leitura/Escreita em Sockets



Let's Code!



Exercícios

Praticando modelo
cliente-servidor com socket



emerson@paduan.pro.br

Exercício 1

Altere o servidor para que ele retorne ao cliente uma confirmação de que recebeu a mensagem enviada pelo cliente.

Altere o cliente para receber uma confirmação do servidor a cada mensagem enviada.

Exercício 2

Altere o cliente para enviar ao servidor várias mensagens digitadas pelo usuário, durante a mesma conexão. O cliente será encerrado ao receber a mensagem 'exit' digitada pelo usuário.

Altere o servidor para que ele fique recebendo diversas mensagens do cliente até que o cliente envie a mensagem "exit".

Exercício 3

Altere o Servidor para que ele não encerre depois de terminar a conexão com um cliente. Ou seja, após o cliente encerrar, ele volta para o estado de aguardar uma nova conexão.

Para encerrar definitivamente o Servidor pode ser utilizada a mensagem "fechar" vinda do cliente.