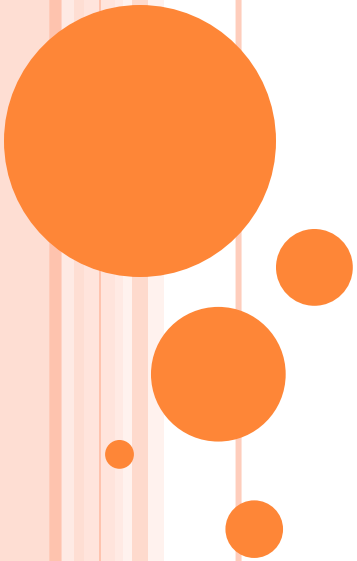


Tecnologia Java para Sockets

Lucas Coutinho de S. Oliveira

Inter-Process Communication (IPC)

Inter-Process Communication (IPC)

- O que é?
  - "[...] **inter-process communication (IPC)** is a set of methods for the exchange of data among multiple threads in one or more processes. Processes may be running on one or more computers connected by a network." [3]

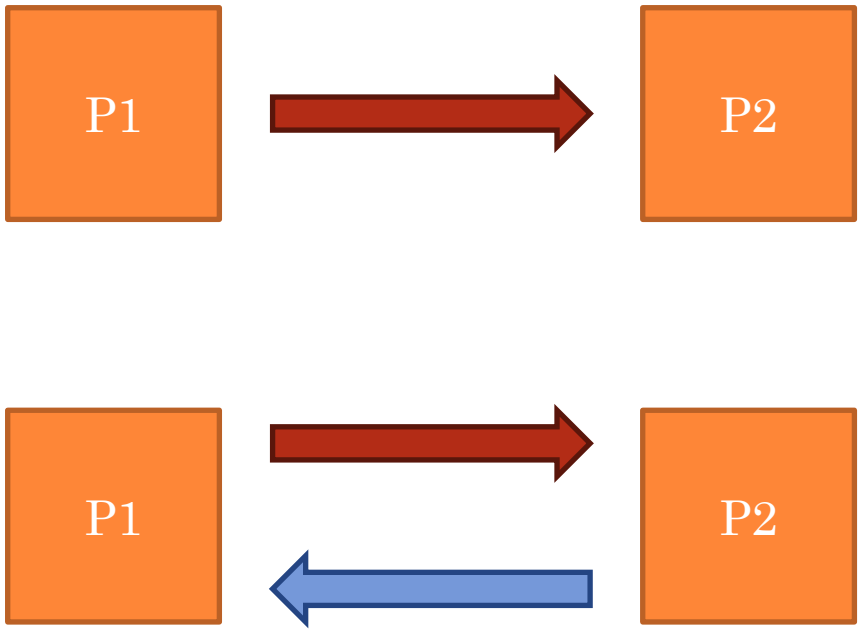
  - "IPC é um conjunto de métodos para troca de informações entre múltiplas threads em um ou mais processos. Processos podem estar executando em um ou mais computadores conectados por um rede." (tradução livre)

Inter-Process Communication (IPC)

- É considerado, por muitos, o *backbone* da computação distribuída;
- Permite que processos separados comuniquem-se entre si para executar um tarefa comum;

Inter-Process Communication (IPC)

- A troca de informações podem ser:
  - Simplex
  - Duplex



Inter-Process Communication (IPC)

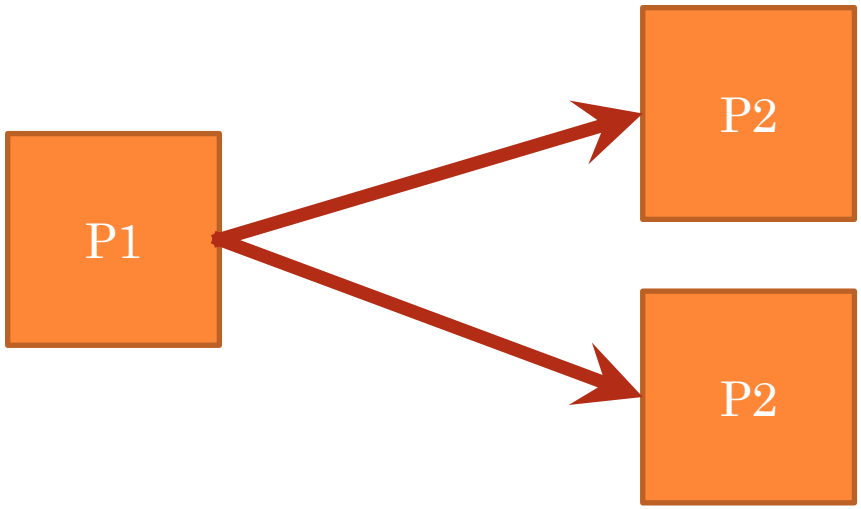
- E, ainda podem ser classificadas como:
  - Unicast

    O processo **P1** troca dados com apenas um processo **P**;

Inter-Process Communication (IPC)

- E, ainda podem ser classificadas como:
  - Multicast

    O processo P1 troca dados, simultaneamente com vários processos[1];



[1] **Não são várias comunicações unicasts;**

Inter-Process Communication (IPC)

- Protocolos
  - São um conjunto de regras que precisam ser respeitadas para que a comunicação aconteça;

Inter-Process Communication (IPC)

- Exemplos:
  - Pipes (nomeados e unidirecionais);
  - Fila de mensagens;
  - Memória compartilha;
  - RPC (Remote Procedure Call);
  - Socket;

Sockets

Sockets

- Foram definidos originalmente no Berkeley Unix 4.x;
- São um *elo de comunicação, bidirecional, entre dois processos em execução distribuídos em uma rede.*[6]

Sockets

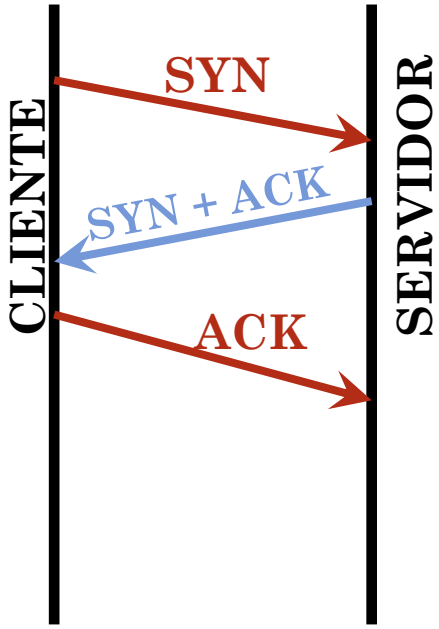
- Endereçamento:
  - Endereço IP + Porta
- Portas
  - É um identificador (16 bits) de buffer associado a um processo;
  - 65.536 portas possíveis;

Sockets

- Protocolos (Camada de transporte):
  - TCP (Transmission Control Protocol);
  - UDP (User Datagram Protocol);

Sockets

- Protocolos (Camada de transporte):
  - TCP (Transmission Control Protocol);
    - Orientado a conexão (3-way handshake);
    - Confiabilidade;
    - Full Duplex;
    - Entrega ordenada;
    - Controle de Fluxo;



Sockets

- Protocolos (Camada de transporte):
  - UDP (User Datagram Protocol);
    - Não orientado a conexão;
    - Não confiável;
    - Multicast; (1 → N)

Sockets em Java

Sockets em Java

- Introduzido no JDK 1.0 (1996)
  - Primeira versão lançada;
  - Desenvolvido pela Sun Microsystems;
  - Mantido pela Oracle Corporation (2010);
- O pacote **java.net** fornece uma API para o uso de sockets;
- Fornece suporte a transmissão **unicast** e **multicast**;

Sockets em Java

- Unicast
  - Socket
    - API de cliente TCP;
    - Utilizada para se conectar a um servidor remoto;
    - Envia dados através de um stream;
  - ServerSocket
    - API de servidor TCP;
    - Utilizada para receber conexões de um host remoto
    - Recebe os dados através de um stream;

Sockets em Java

- Servidor TCP (Exemplo de uso)

```
String clientSentence;
String capitalizedSentence;
System.out.println("Ativando o Servidor...");
ServerSocket welcomeSocket = new ServerSocket(6789);
while (true) {
    System.out.println("Aguardando conexão com o Cliente");
    Socket connectionSocket = welcomeSocket.accept();
    BufferedReader inFromClient
            = new BufferedReader(new InputStreamReader(connectionSocket.getInputStream())
            );
    DataOutputStream outToClient
            = new DataOutputStream(connectionSocket.getOutputStream());
    clientSentence = inFromClient.readLine();
    System.out.println("Frase recebida do Cliente: " + clientSentence);
    capitalizedSentence = clientSentence.toUpperCase() + '\n';
    outToClient.writeBytes(capitalizedSentence);
    System.out.println("Frase enviada para o Cliente: " + capitalizedSentence);

}
```

Sockets em Java

- Cliente TCP (Exemplo de uso)

```
String sentence;
String modifiedSentence;
BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));

//Aqui deve-se escolher o endereço do servidor. Para rodar na mesma máquina,
//utiliza-se o endereço de loopback (IP = 127.0.0.1)
System.out.println("Ativando o Cliente...");
Socket clientSocket = new Socket("127.0.0.1", 6789);

DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());
BufferedReader inFromServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()
));
System.out.println("Digite sua frase: ");
sentence = inFromUser.readLine();
System.out.println("Enviando dados para o Servidor...");
outToServer.writeBytes(sentence + '\n');
modifiedSentence = inFromServer.readLine();
System.out.println("Frase recebida do Servidor: " + modifiedSentence);
```

Sockets em Java

- Unicast
  - DatagramSocket
    - API de cliente/servidor UDP;
    - Envia/Recebe dados através de pacotes de datagrama;

Sockets em Java

- Servidor UDP (Exemplo de uso)

```
DatagramSocket serverSocket = new DatagramSocket(9876);
byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];

while (true) {
    DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
    serverSocket.receive(receivePacket);
    String clientSentence = new String(receivePacket.getData());
    System.out.println("Frase recebida do Cliente: " + clientSentence);
    InetAddress IPAddress = receivePacket.getAddress();
    int port = receivePacket.getPort();
    String capitalizedSentence = clientSentence.toUpperCase();
    System.out.println("Frase enviada para o Cliente: " + capitalizedSentence);
    sendData = capitalizedSentence.getBytes();
    DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
            IPAddress, port);
    serverSocket.send(sendPacket);
}
```

Sockets em Java

- Cliente UDP (Exemplo de uso)

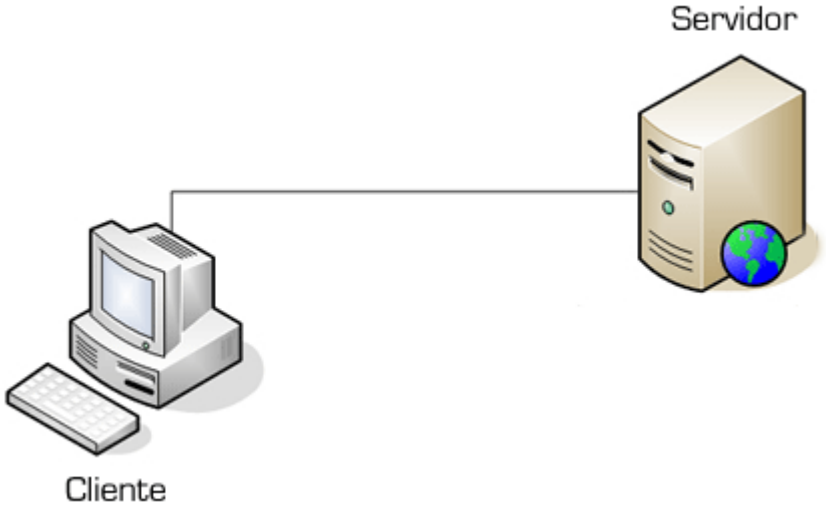

```
BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
DatagramSocket clientSocket = new DatagramSocket();
InetAddress IPAddress = InetAddress.getByName("127.0.0.1");
byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];
System.out.println("Digite sua frase: ");
String sentence = inFromUser.readLine();
sendData = sentence.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress,
        9876);
clientSocket.send(sendPacket);
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
clientSocket.receive(receivePacket);
String modifiedSentence = new String(receivePacket.getData());
System.out.println("Frase recebida do Servidor: " + modifiedSentence);
clientSocket.close();
```

Sockets em Java

- Unicast
  - DatagramSocket
    - API de cliente/servidor UDP;
    - Envia/Recebe dados através de pacotes de datagrama;

Sockets em Java

- Unicast
  - Arquitetura: Cliente-Servidor



Sockets em Java

- Multicast
  - Útil em aplicações groupware;
    - Ex: Vídeo Conferencia;
  - Utiliza o conceito de grupos, formados por diversos processos (***multicast group***);
  - Cada processo no grupo pode enviar mensagem para todos os outros;

Sockets em Java

- Multicast
  - Faixa de endereços:
    - 224.0.0.0 até 239.255.255.255
  - Reservas
    - 224.0.0.0 Base Address (Reserved);
    - 224.0.0.1 Todos os sistemas (rede IP multicast);
    - 224.0.0.2 Todos os roteadores (rede IP multicast);

Sockets em Java

- Multicast
  - A Multicast API suporta algumas operações primitivas:
    - Join – Adiciona um processo, ou um grupo de processos, a um grupo multicast;
    - Leave – Remove um processo do grupo;
    - Send – Envia mensagem para todos os processos de um grupo;
    - Receive - Permite ao processo receber mensagens;

Sockets em Java

- Multicast
  - Classes da Multicast API
    - InetAdress
    - DatagramPacket
    - DatagramSocket
    - MulticastSocket (estende DatagramSocket)

Sockets em Java

- Receiver Multicast (Exemplo de uso)

```
InetAddress group = InetAddress.getByName("224.0.0.1");
InetAddress group = InetAddress.getByName("224.0.0.1");
MulticastSocket multicastSock = new MulticastSocket(3456);
multicastSock.joinGroup(group);
byte[ ] buffer = new byte[100];
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
multicastSock.receive(packet);
System.out.println(new String(buffer));
multicastSock.close( );
```

Sockets em Java

- Sender Multicast (Exemplo de uso)

```
InetAddress group = InetAddress.getByName("224.0.0.1");
MulticastSocket multicastSock = new MulticastSocket(3456);
String msg = "Hello How are you?";
DatagramPacket packet = new DatagramPacket(msg.getBytes( ), msg.length( ), group,
3456);
multicastSock.send(packet);
multicastSock.close( );
```

WebSocket

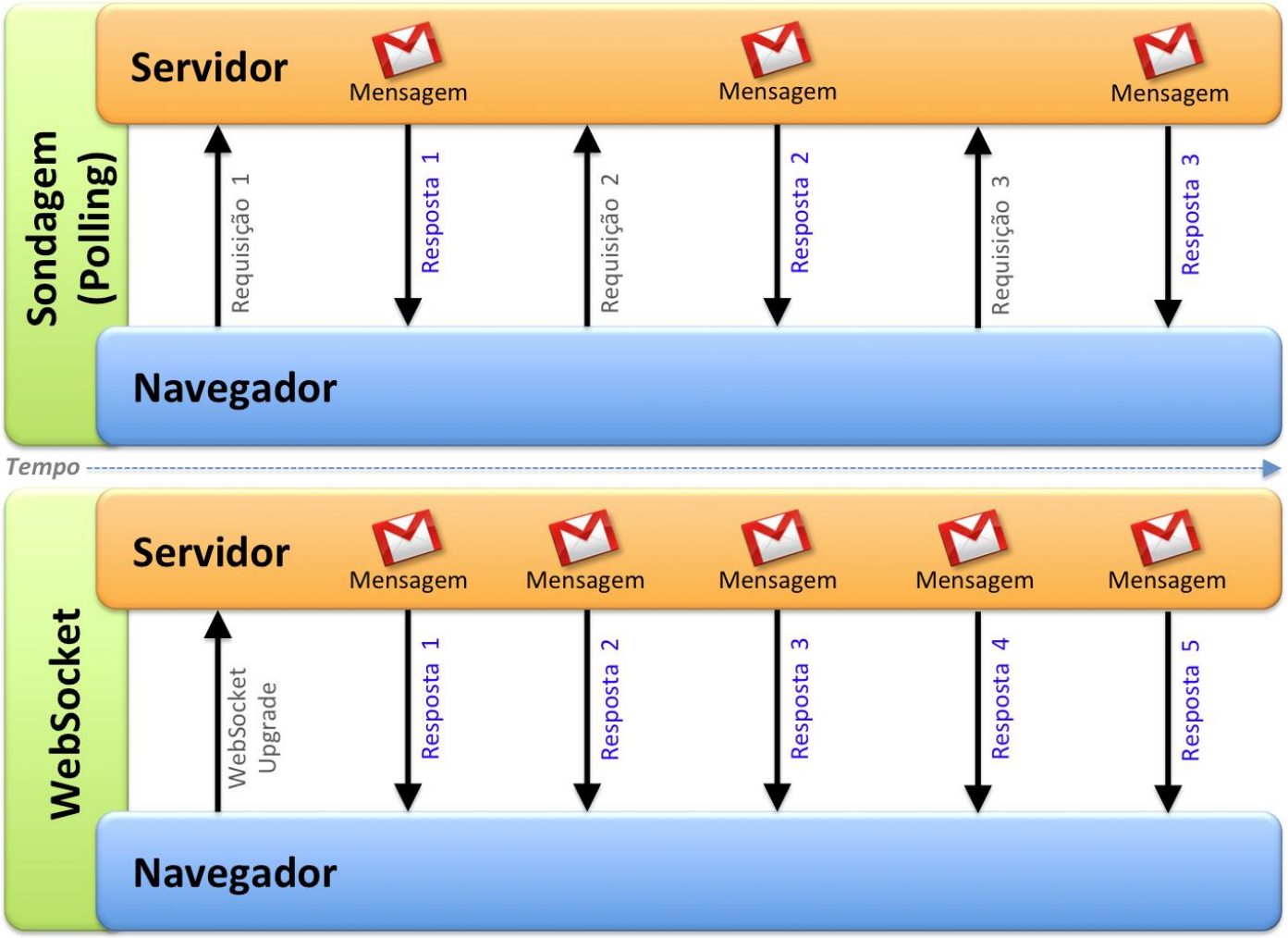
WebSocket

- Introdução:
  - Historicamente, a criação de aplicações web que precisam de comunicação bidirecional entre cliente e servidor tem exigido um abuso do HTTP efetuando consultas, para manter as informações constantemente atualizadas. (Por exemplo, chats e aplicativos de jogos)
  - Esse abuso resulta em uma variedade de problemas. São eles:
    - Necessidade de fazer Polling (sondagem) para manter atualização;
    - Cada mensagem cliente-servidor possui um cabeçalho HTTP, gerando um overhead;
    - Scripts (client-side) precisam manter mapeamento de conexões de entrada e saída para acompanhar as respostas;
    - Etc..

WebSocket

- Introdução
  - Uma solução simples, seria utilizar conexões TCP bidirecionais;
  - E é exatamente isso que o **Protocolo WebSocket** provê;

WebSocket


Sondagem (Polling)
Servidor
Mensagem
Mensagem
Mensagem
Requisição 1
Resposta 1
Requisição 2
Resposta 2
Requisição 3
Resposta 3
Navegador
Tempo
WebSocket
Servidor
Mensagem
Mensagem
Mensagem
Mensagem
Mensagem
WebSocket Upgrade
Resposta 1
Resposta 2
Resposta 3
Resposta 4
Resposta 5
Navegador

WebSocket

- RFC6455
  - O **protocolo WebSocket** foi definido em Dezembro 2011;
  - Especificação foi produzida pela Internet Engineering Task Force (IETF);

- API WebSocket
  - Está sendo padronizada pelo World Wide Web Consortium (W3C);

WebSocket

- Funcionamento
  - Estabelecendo a conexão (promoção HTTP → WebSocket)

WebSocket

- Funcionamento

```
GET /chat HTTP/1.1
  Host: server.example.com
  Upgrade: websocket
  Connection: Upgrade
  Sec-WebSocket-Key: jUllmHNhbWXtZSClA33jQQ==
  Origin: http://example.com
  Sec-WebSocket-Protocol: chat, superchat
  Sec-WebSocket-Version: 13
```

Handshake: promoção HTTP → WebSocket

Cliente → Servidor

WebSocket

- Funcionamento

```
HTTP/1.1 101 Switching Protocols
  Upgrade: websocket
  Connection: Upgrade
  Sec-WebSocket-Accept: i3pPLGGifxAQ1kYZyzhZRbK+xOo=
  Sec-WebSocket-Protocol: chat
```

Handshake: promoção HTTP → WebSocket

Servidor → Cliente

WebSocket

- Java API for WebSocket (JSR 356)
  - Está presente no JavaEE 7.0
  - Consiste nos pacotes:
    - **Javax.websocket.server**
      - Contem annotations, classes e interfaces para criar e configurar os servers endpoints;
    - **Java.websocket**
      - Contem annotations, classes, interfaces e exceptions comuns a clientes e servidores;

WebSocket

- Java API for WebSocket (JSR 356)
  - WebSocket Endpoint
    - Programmatic Endpoint
    - Annotated Endpoint

WebSocket

- Java API for WebSocket (JSR 356)
  - WebSocket Endpoint (Criação)

    1. **Criar a classe Endpoint;**
    2. **Implementar o métodos do ciclo de vida do endpoint;**
    3. **Adicionar a lógica de negócios;**
    4. **Deploy da aplicação web;**

WebSocket

- Java API for WebSocket (JSR 356)
  - WebSocket Endpoint (Ciclo de Vida)

Annotation	Evento
@OnOpen	Inicio da conexão
@OnMessage	Mensagem recebida
@OnErro	Erro de conexão
@OnClose	Término da conexão

Referências

[1] http://143.132.8.23/cms/tues/docs/ComputerNetworks/Java-Socket-Programming-Manual.pdf

[2] http://download.java.net/jdk7/archive/b123/docs/api/java/net/DatagramSocket.html

[3] http://en.wikipedia.org/wiki/Interprocess_communication

[4] http://gsd.di.uminho.pt/teaching/misd/2007/od/sockets.pdf

[5] http://pt.wikipedia.org/wiki/Comunica%C3%A7%C3%A3o_entre_processos

[6] http://www.cs.uic.edu/~troy/fall04/cs441/drake/sockets.html

[7] http://www.devmedia.com.br/via-de-mao-dupla-com-websockets-revista-java-magazine-117/28281

[8] http://www.heldervaldez.com/redes-computadores/521-tcp-e-udp-diferencas-entre-protocolos.html

[9] http://www.html5rocks.com/pt/tutorials/websockets/basics/

[10] http://www.inetdaemon.com/tutorials/basic_concepts/communication/index.shtml