Comunicação entre Processos

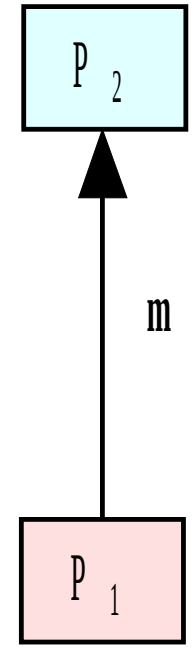
Comunicação entre Processos

- Sistemas Operacionais fornecem mecanismos para comunicação entre processos (IPC), tal como filas de mensagens, semáfaros e memória compartilhada.
- Sistemas de computação distribuída utilizam estes mecanismos para fornecer uma interface de programação da aplicação (API) que permita a comunicação entre processos ser programada em alto nível de abstração.
- Computação distribuída requer troca de informação entre processos independentes.

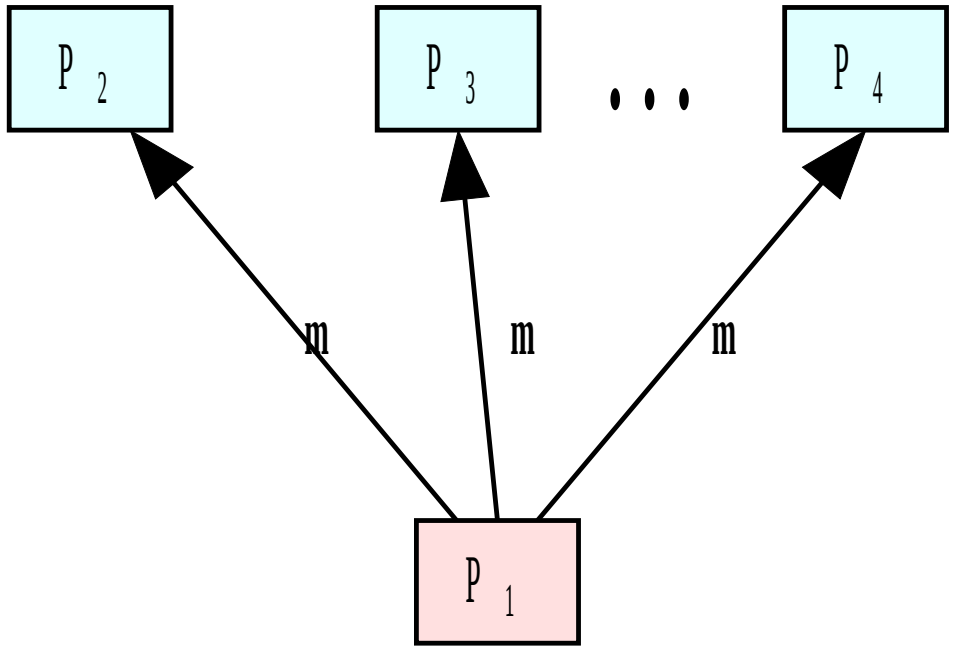
IPC – unicast and multicast

- Em computação distribuída, dois ou mais processos envolvidos em IPC através de um protocolo que ambos comcordam. Um processo pode ser o enviante em alguns pontos durante o protocolo um recebedor durante outros.
- Quando a comunicação é de um processo para outro o IPC é chamado de ***unicast***. Quando a comunicação é de um processo para um grupo de processos, o IPC é chamado de ***multicast***.

Unicast vs. Multicast

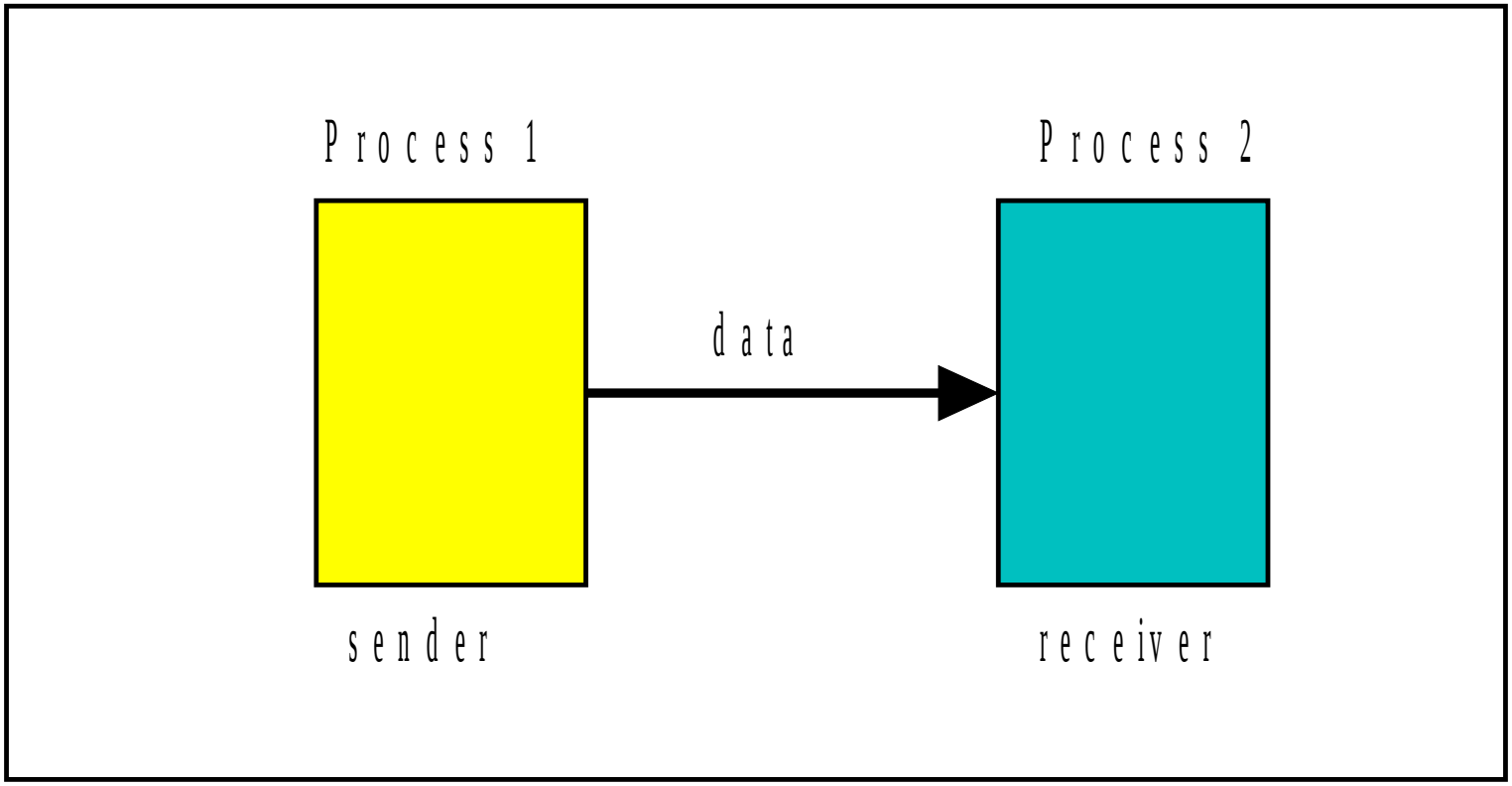


unicast



multicast

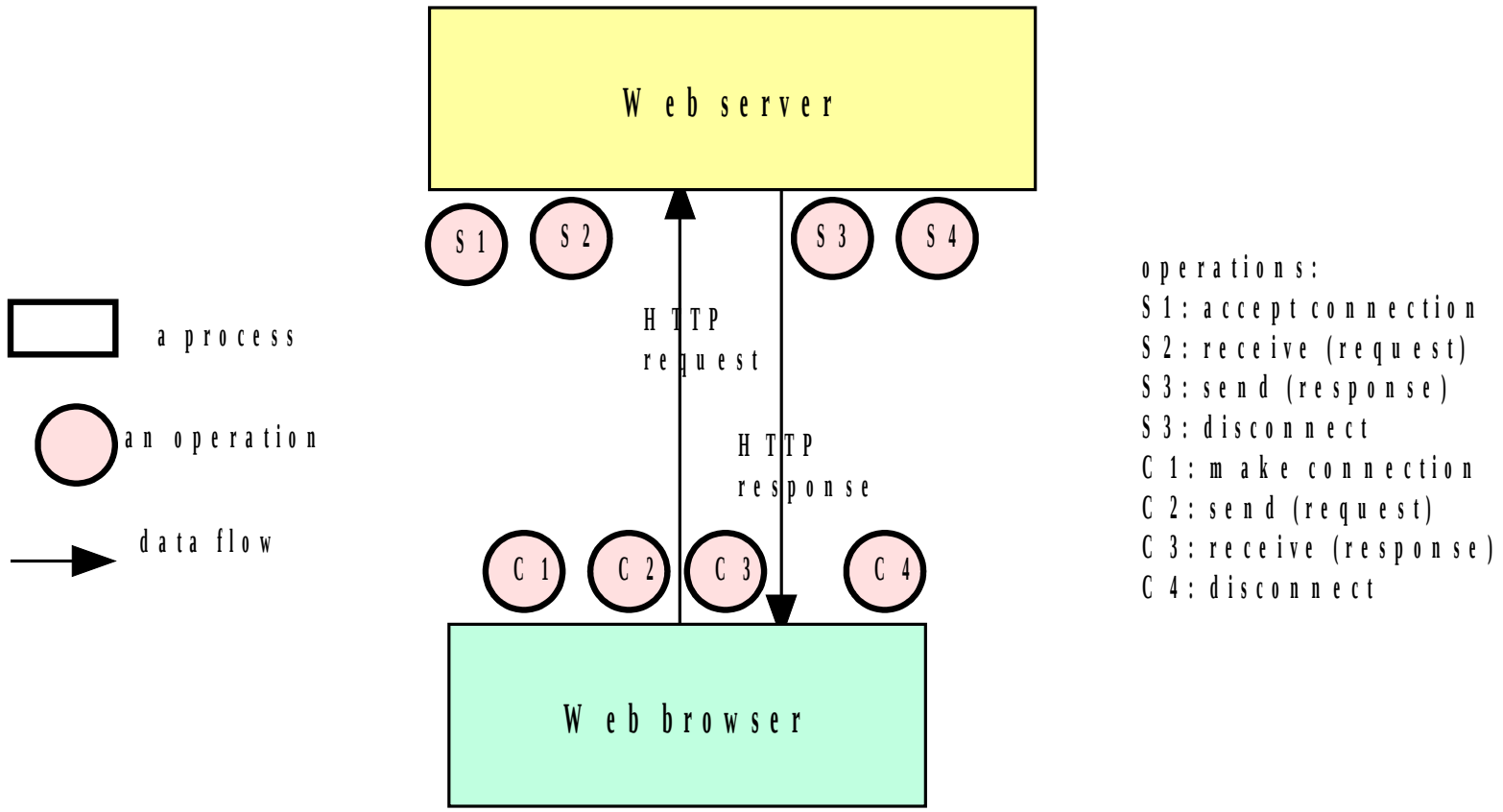
Comunicação entre Processos em computação distribuída



Operações básicas fornecidas em uma API para comunicação entre processos

- Receive ([sender], buffer_da_mensagem)
- Connect (endereço sender, endereço receiver), para comunicação orientada a conexão
- Send ([receiver], mensagem)
- Disconnect (identificador da conexão), para comunicação orientada a conexão.

IPC em HTTP


Web server
S1
S2
S3
S4
HTTP request
HTTP response
C1
C2
C3
C4
Web browser
a process
an operation
data flow
operations:
S1: accept connection
S2: receive (request)
S3: send (response)
S3: disconnect
C1: make connection
C2: send (request)
C3: receive (response)
C4: disconnect

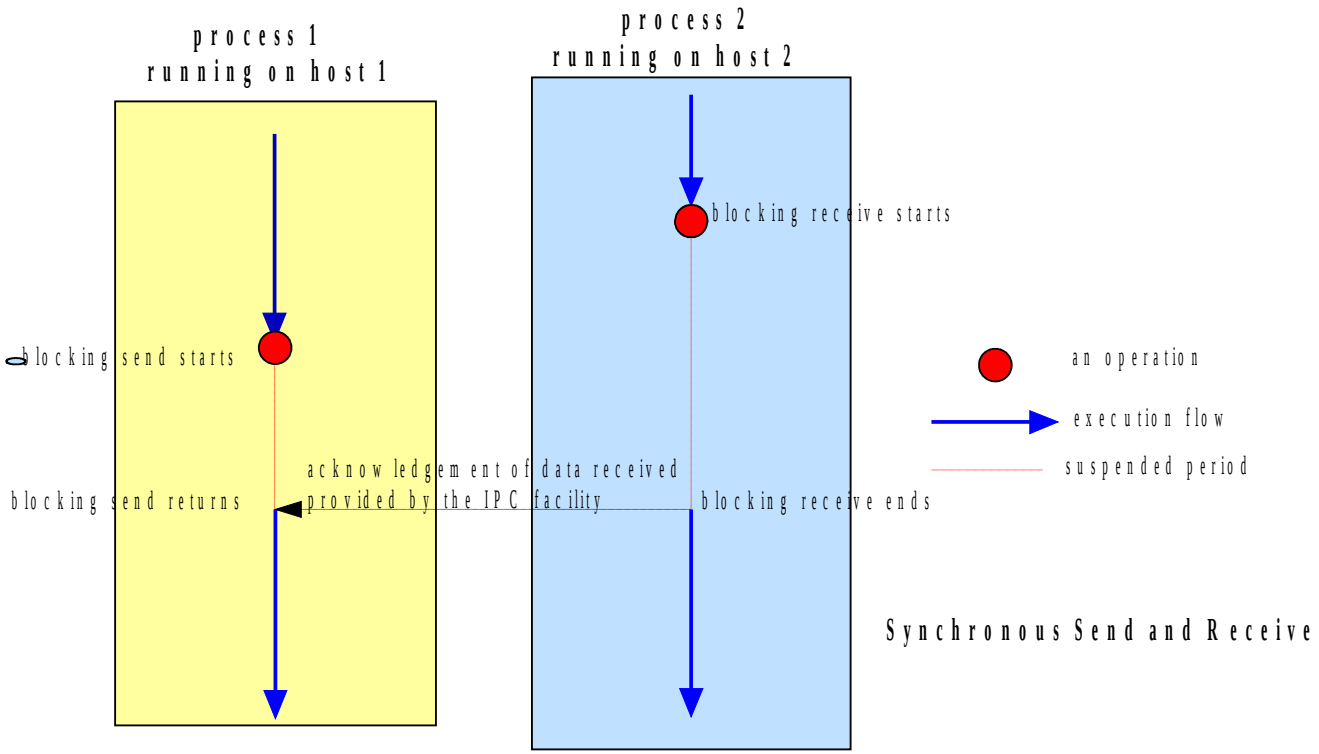
Sincronização de Evento

- Comunicação entre processos requer que 2 processo sincronizem suas operações: um lado envia e o outro recebe até que todos os dados tenha sido enviados e recebidos.
- Ideal, a operação send inicia antes da operação receive começar.
- Na prática, a sincronização requer suporte do sistema.

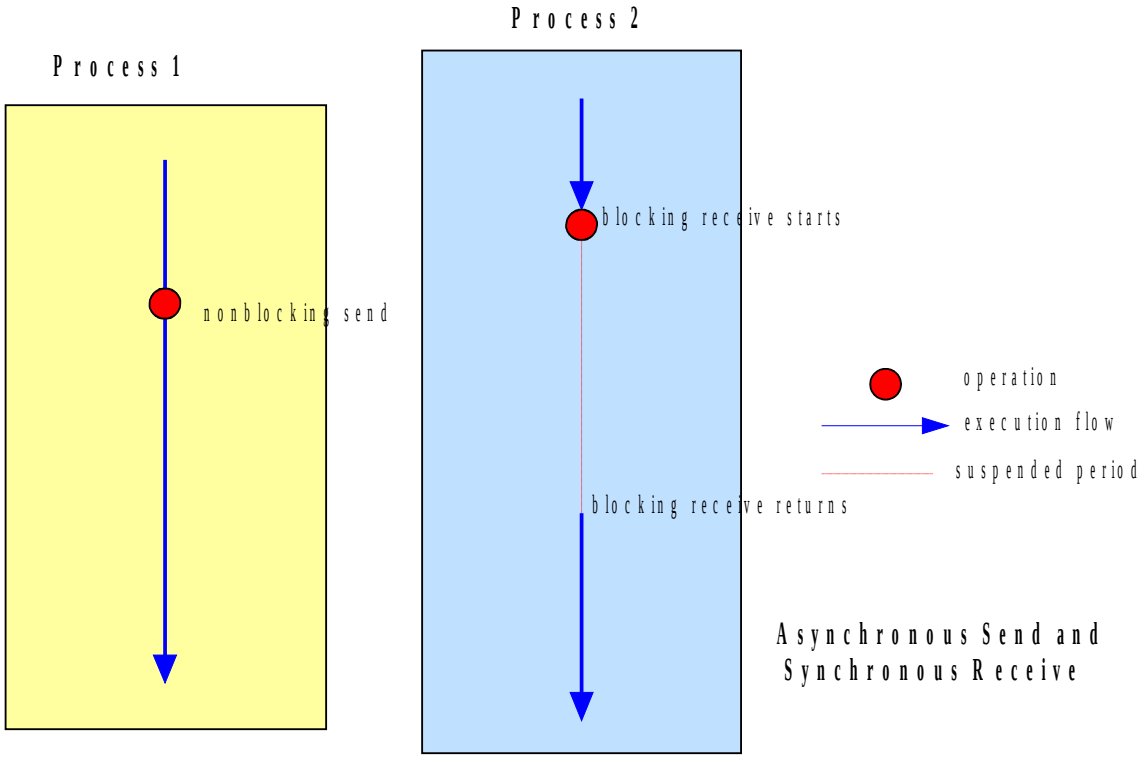
Comunicação Síncrona X Assíncrona

- **As operações IPC podem fornecer a sincronização necessária usando *bloqueamento*. Uma operação bloqueante emitida por um processo irá bloquear o processamento do processo até a operação ser completada.**
- **Alternativamente, operações IPC podem ser assíncronas ou *não bloqueantes*. Uma operação assíncrona emitida por um processo não irá bloquear o processamento do processo. Ao contrário, o processo esta livre para proceder seu processamento e pode opcionalmente ser notificado pelo sistema quando a operação é completada.**

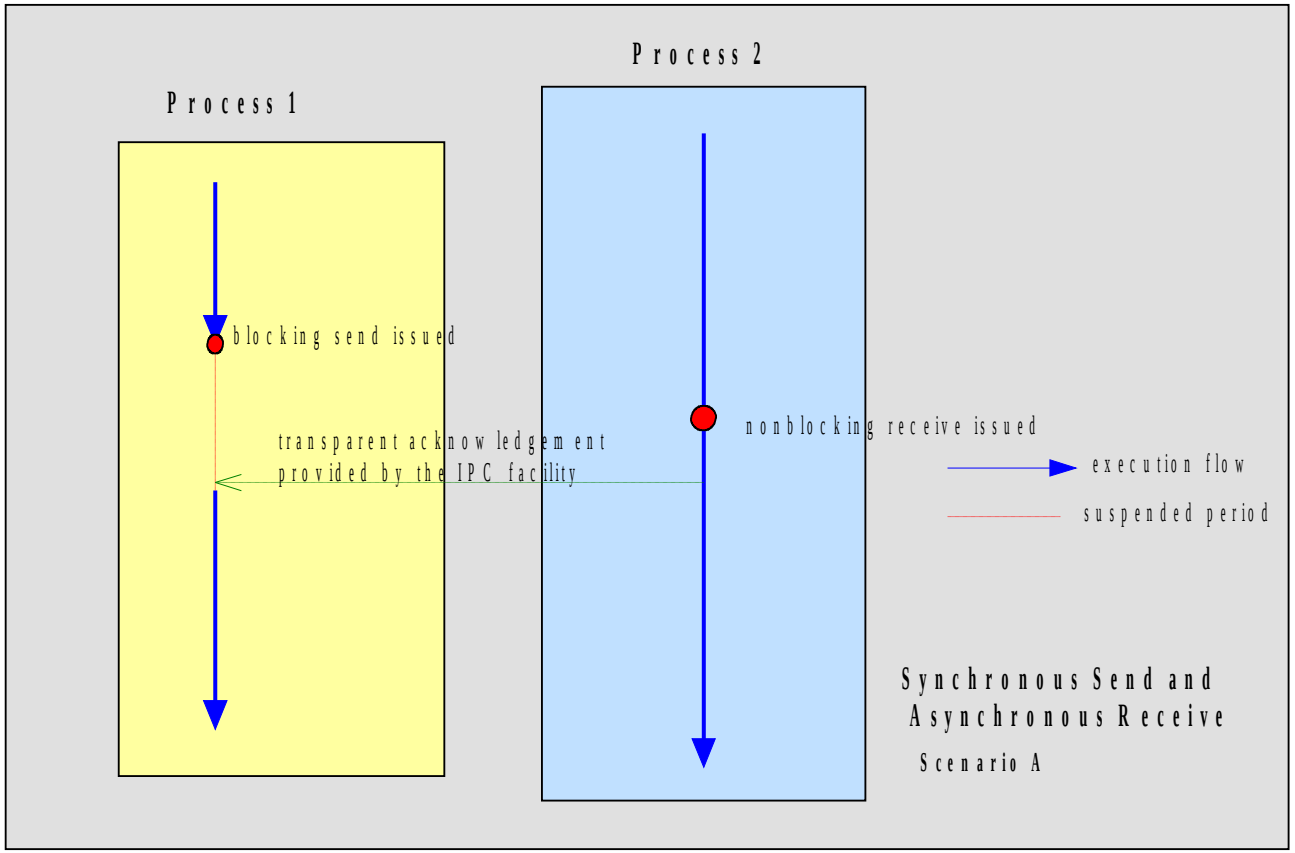
send e receive Síncrono



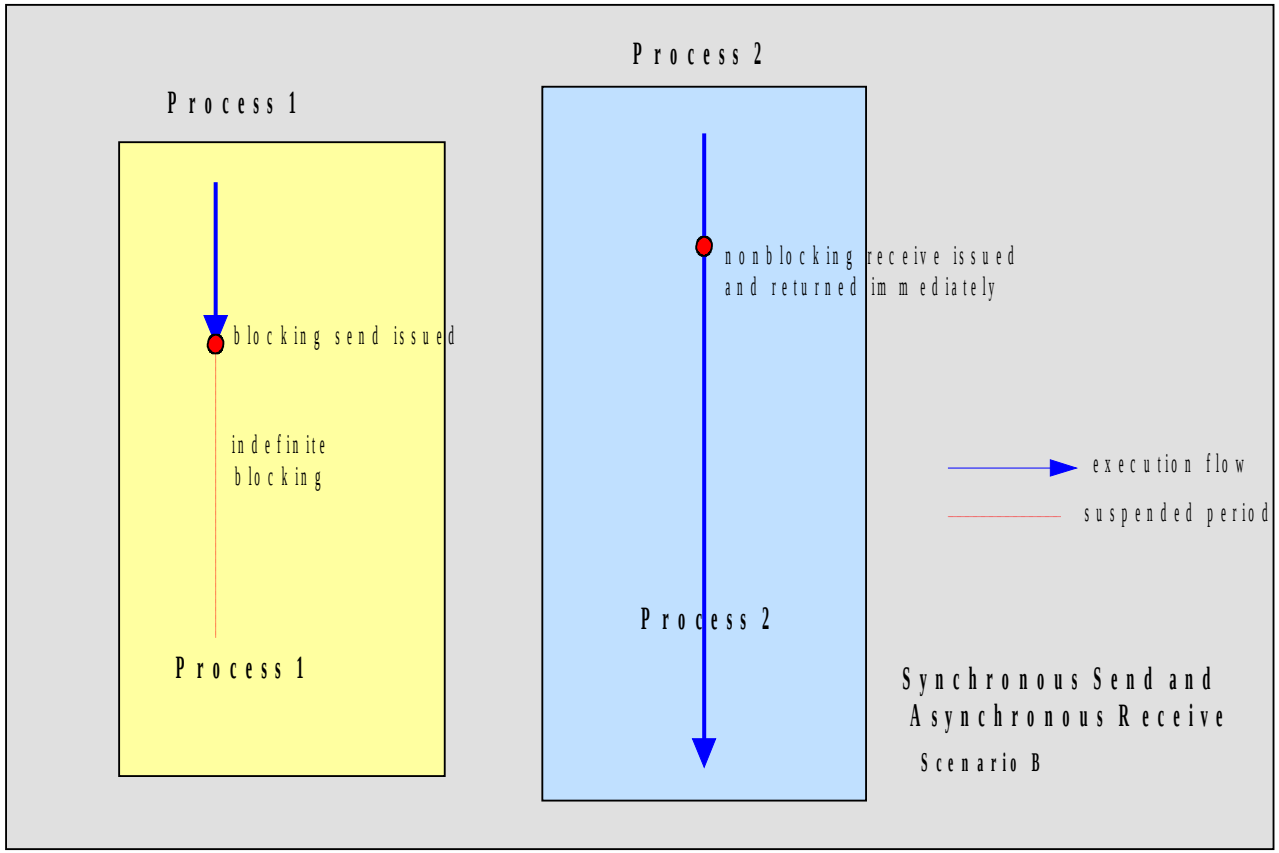
send Assíncrono e receive Síncrono



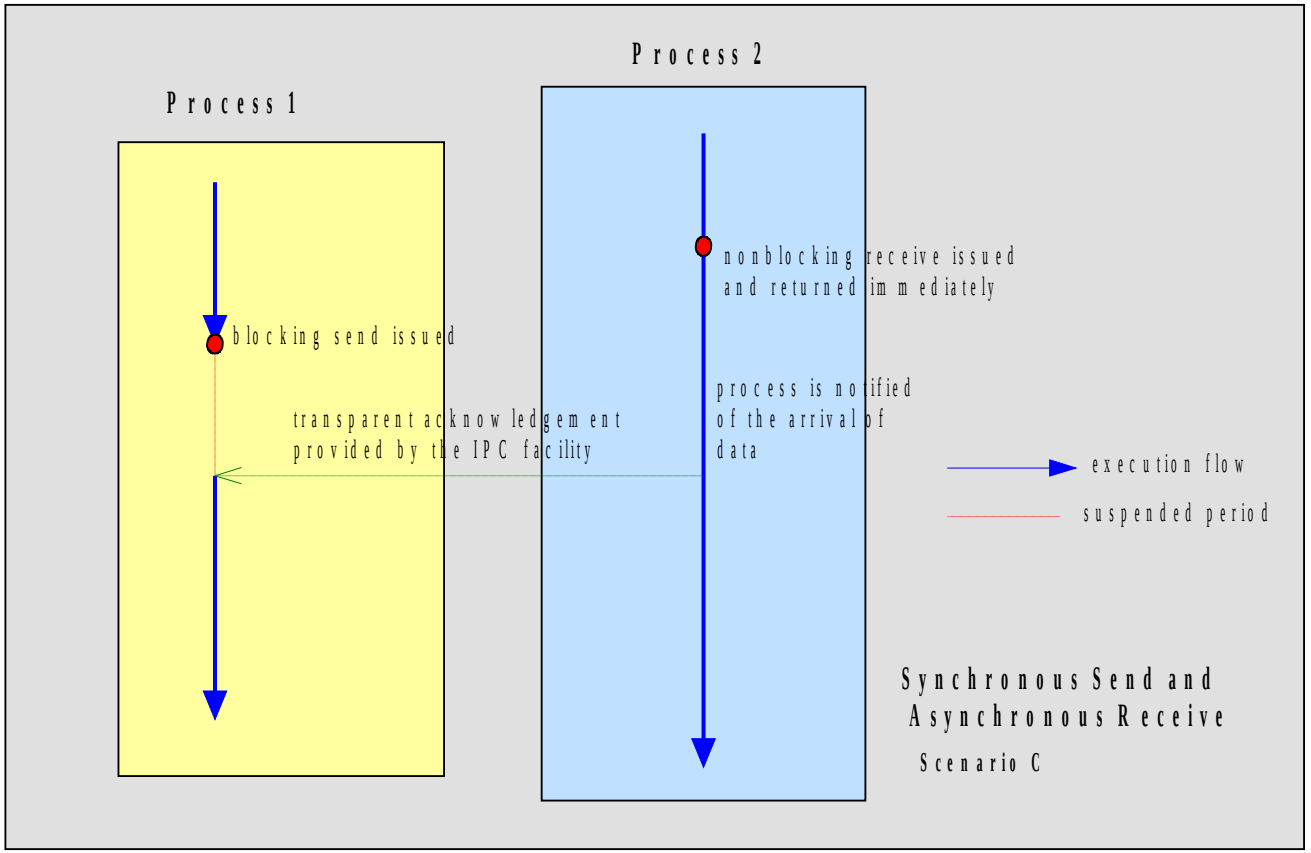
send síncrono e receive assíncrono - 1



send síncrono e receive assíncrono - 2



send síncrono e receive assíncrono - 3



send assíncrono e receive assíncrono

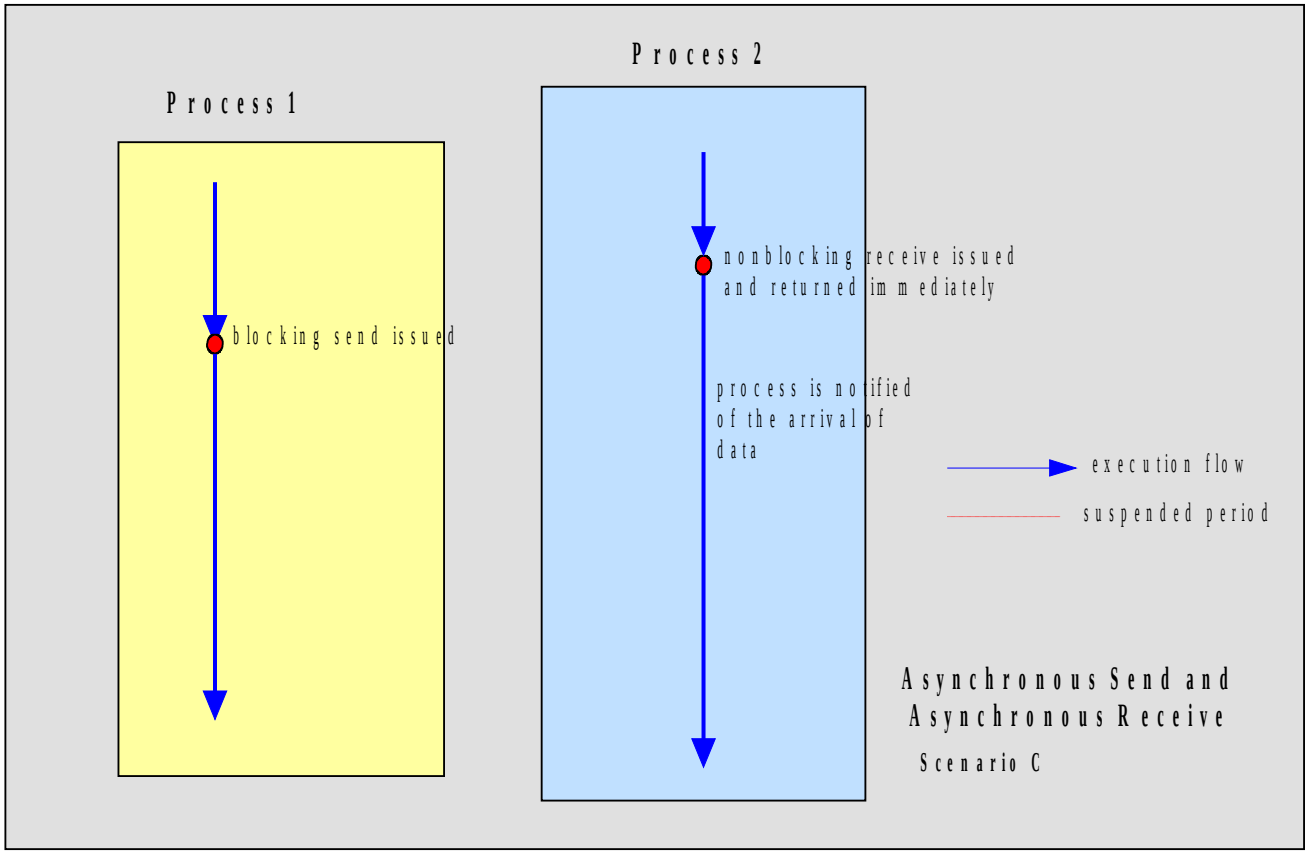
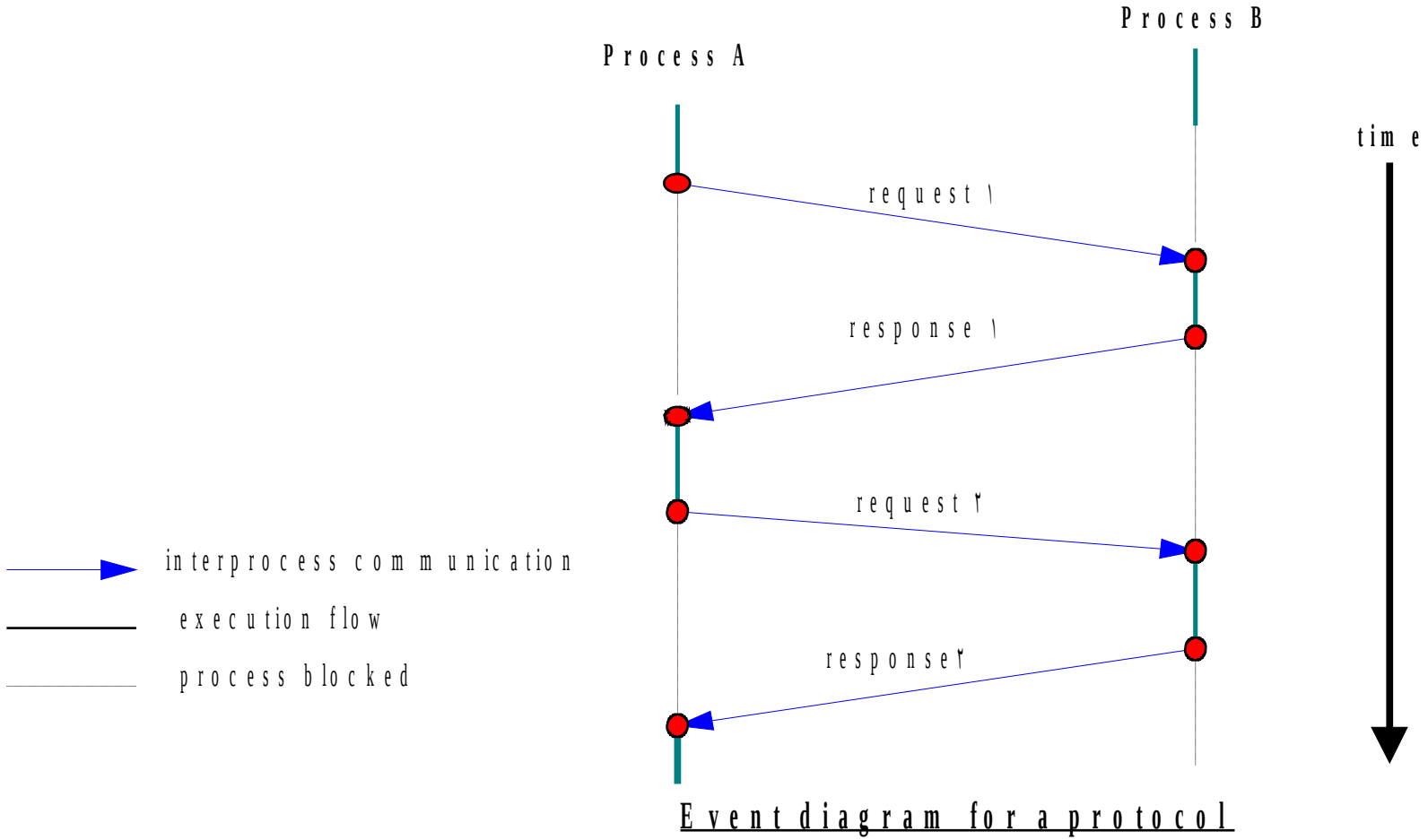


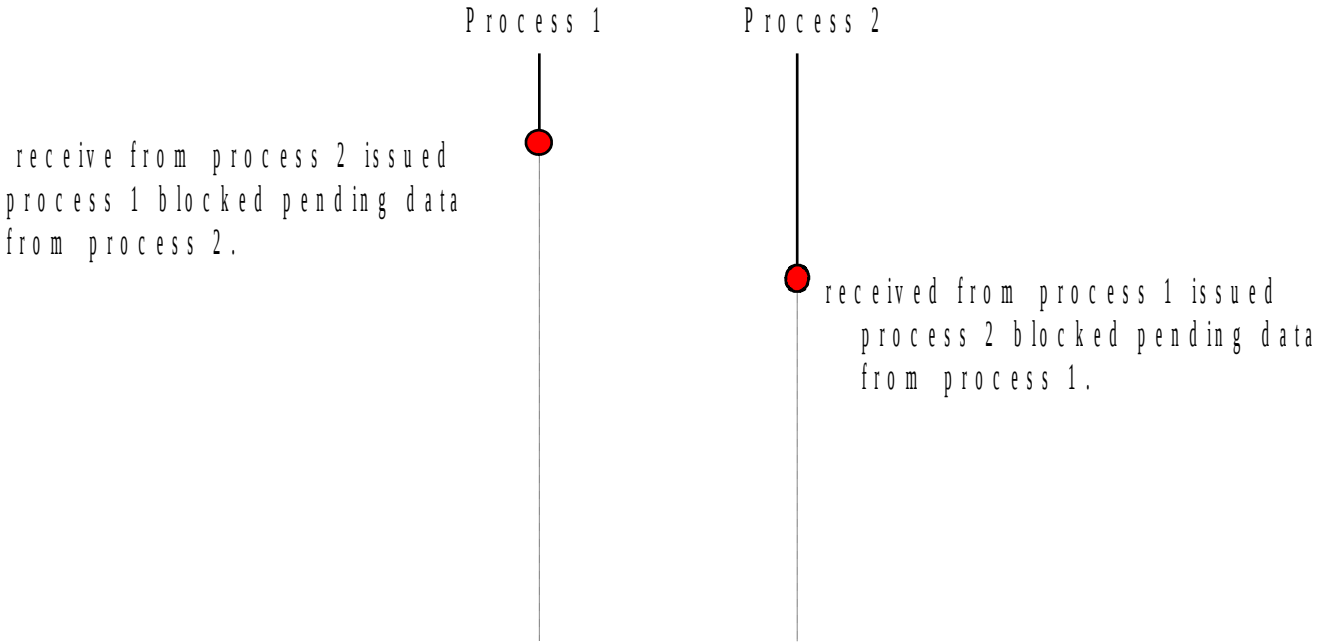
diagrama de eventos



Event diagram for a protocol

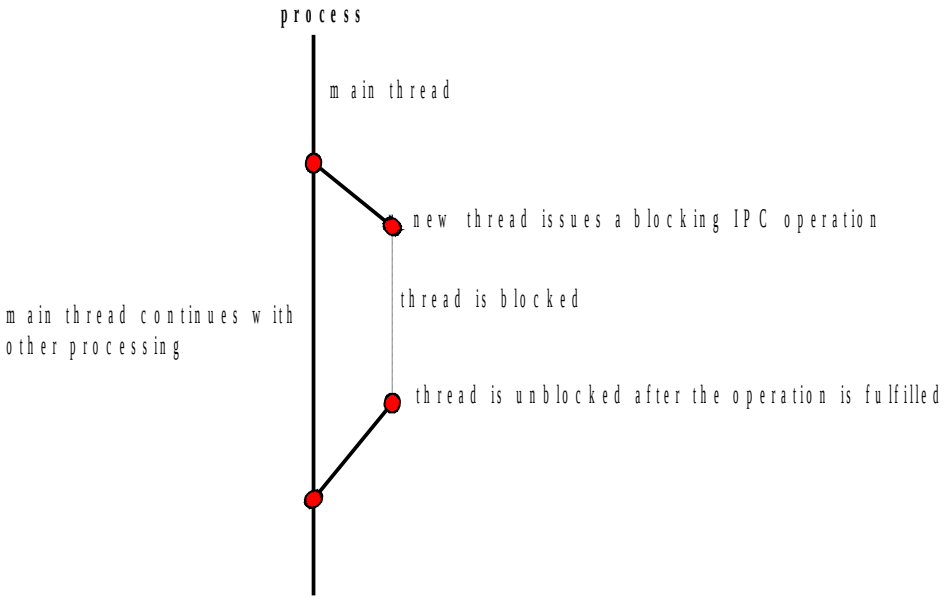
Bloqueios, impasses e timeouts

- **Operações bloqueantes emitidas em sequência errada pode causar *impasses*.**
- **Impasses devem ser evitados. Alternativamente, *timeout* pode ser usado para detectar deadlocks.**



Usando threads para IPC assíncrono

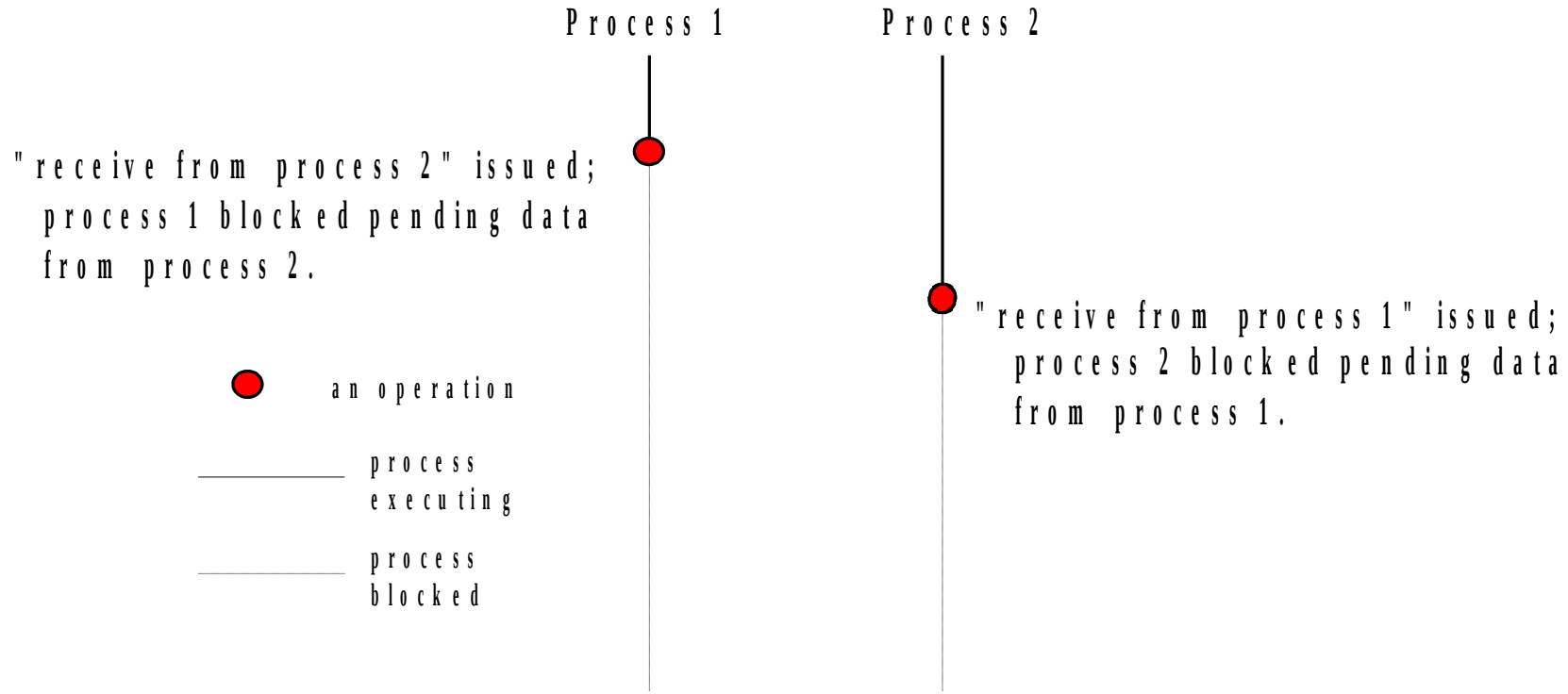
- **Na utilização de uma API IPC, é importante saber quando as operações são síncronas ou assíncronas.**
- **Se apenas operações bloqueantes são fornecidas para send e/ou receive, então é responsabilidade do programador utilizar processos ou threads se operações assíncronas são desejadas.**



Impasses e Timeouts

- Operações connect e receive podem resultar em bloqueio indefinido
- Exemplo, uma requisição bloqueante *connect* pode resultar no bloqueio do processo requisitante que será suspenso indefinidamente se a conexão não for completada ou não puder ser completada como resultado de uma falaha na rede.
- Geralmente não é aceitável para um processo requisitante esperar indefinidamente. Bloqueio indefinido pode ser evitado usando **timeout**.
- Bloqueio indefinido pode também ser causado por um *impasse*

Bloqueio Indefinido devido a impasse (deadlock)



Representação de dados

- Programas -> estruturas de dados, info em msg é seqüencial
- Estruturas de dados devem ser sequencializadas antes da transmissão e reconstruídas na chegada.
- Info em msgs podem ser tipos diferentes de valores, são armazenados de formas diferentes.
- Troca de dados envolve conversão para formato comum antes da trans. e conversão local na recepção.
- ***Data marshalling* é o processo de (i) sequencializar uma estrutura de dados e (ii) converter os dados para uma representação externa.**

Alguns esquemas de representação de dados bem conhecidos são:

Sun XDR

ASN.1 (Abstract Syntax Notation)

XML (Extensible Markup Language)

Protocolos de Codificação dos Dados

Representação Externa (XDR)

5
“Smit”
“h---”
6
“Lond”
“on--”
1934

- msg composta por seqüência de objetos de 4 bytes
- inteiro 1 objeto, string de 4 chars 1 objeto
- arrays, structures, strings: seqüências de bytes com tam. especificado.
- Reduz carga computacional.
- Gasta banda passante.

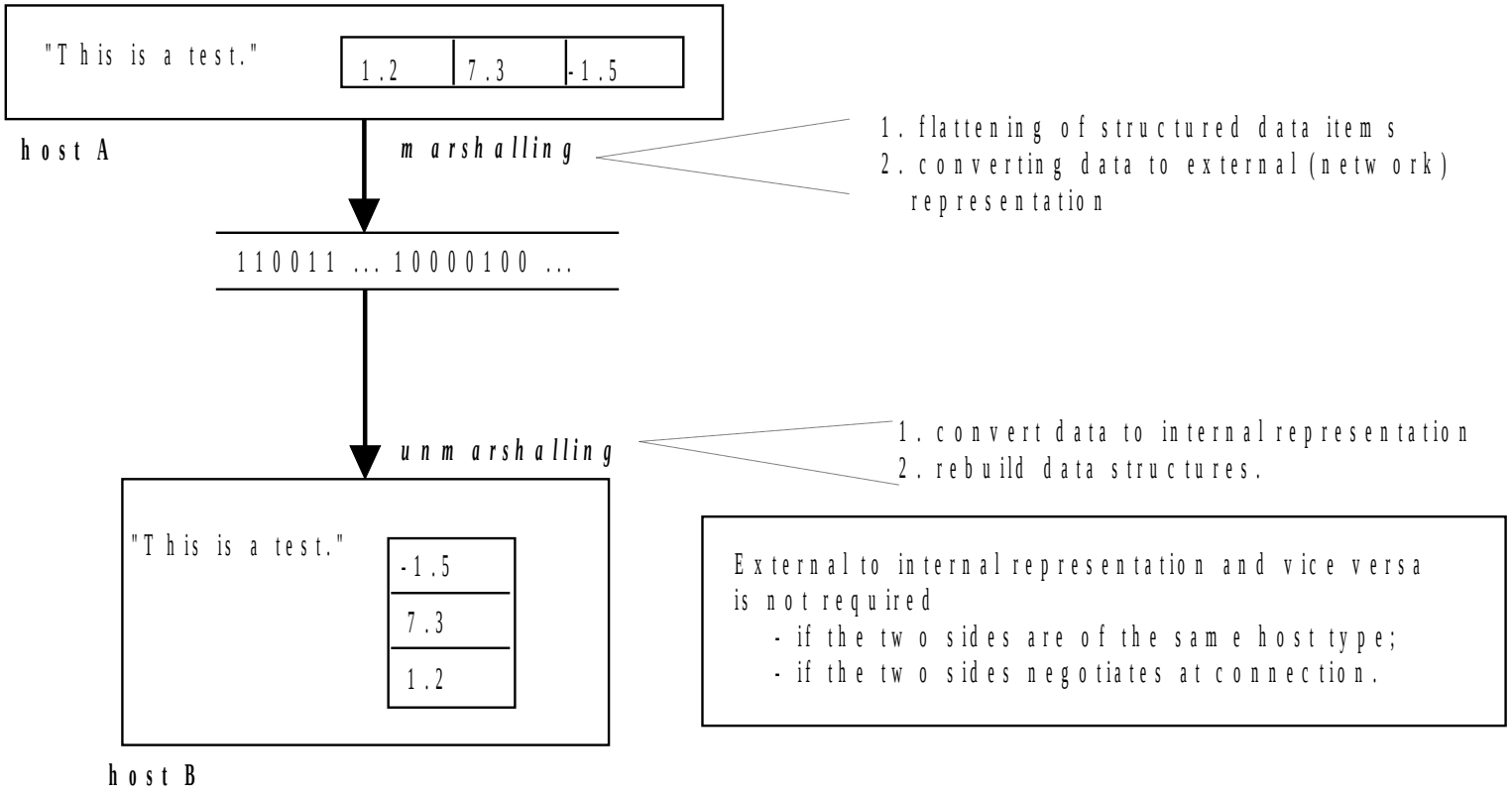
Amostra de um arquivo XML

http://java.sun.com/xml/docs/tutorial/overview/1_xml.html#intro

- **XML linguagem de marcação, tornando padrão troca de dados na Web.**
- **XML sintaxe semelhante a HTML.**
- **Diferente de HTML, tags XML dizem o significado dos dados, ao contrário de como mostra-los.**
- **Exemplo:**

```
<message>
        <to>you@yourAddress.com</to>
        <from>me@myAddress.com</from>
        <subject>XML Is Really Cool</subject>
        <text> How many ways is XML cool? Let me count the
        ways... </text>
</message>
```

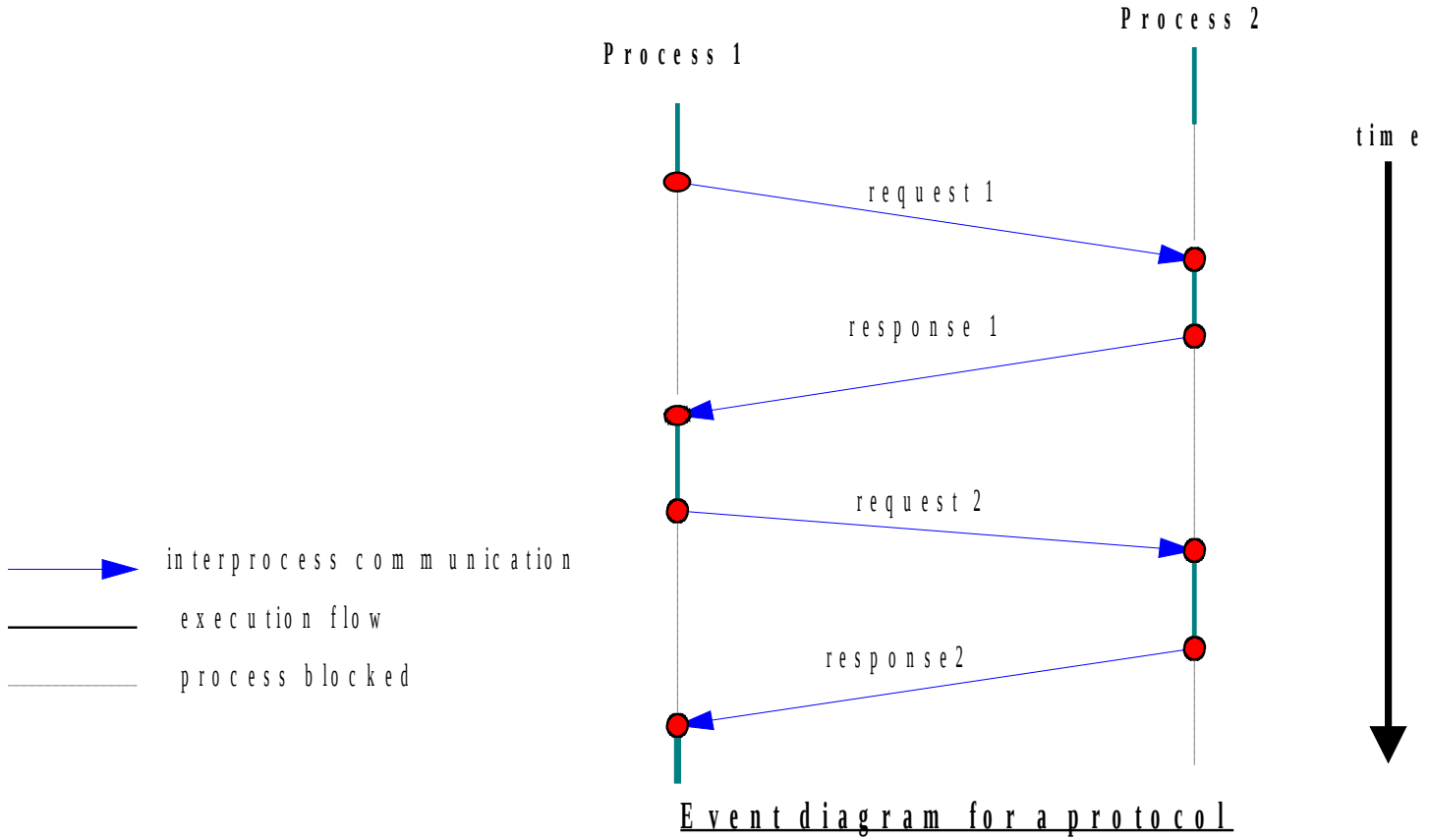
Marshalling



protocolos baseados em texto

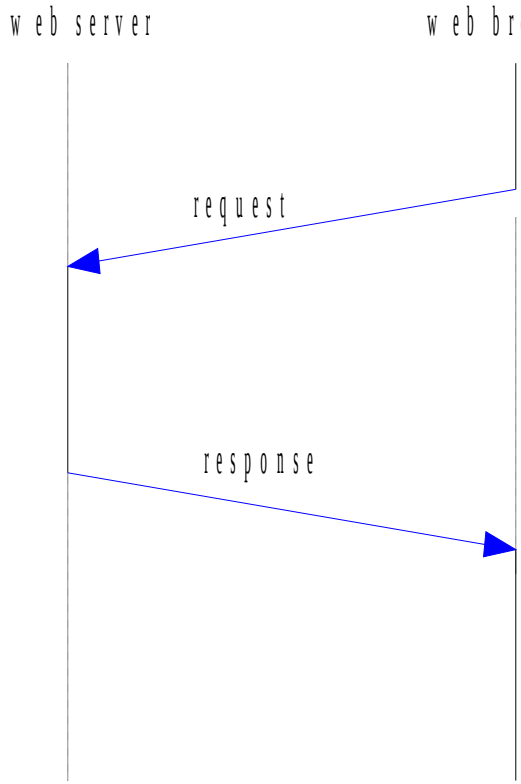
- **Marshalling é mais simples quando os dados trocados são na forma de texto.**
- **Troca de dados em texto tem vantagem adicional que os dados podem ser facilmente analizados em um programa e mostrados. Desta forma, é prática popular protocolos trocar requisições e respostas na forma de strings de caracteres. Tais protocolos são chamados de baseados em texto.**
- **Muitos protocolos são baseados em texto: FTP (File Transfer Protocol), HTTP, and SMTP (Simple Mail Transfer Protocol).**

Diagrama de Evento para uma sessão de protocolo



Event diagram for a protocol

Diagrama de Evento sessão HTTP



request is a message in 3 parts:
- <command> <document adddress> <HTTP version>
- an optional header
- optional data for CGI data using post method

response is a message consisting of 3 parts:
- a status line of the format <protocol><status code><description>
- header information, which may span several lines;
- the document itself.

Diagrama de Sequência

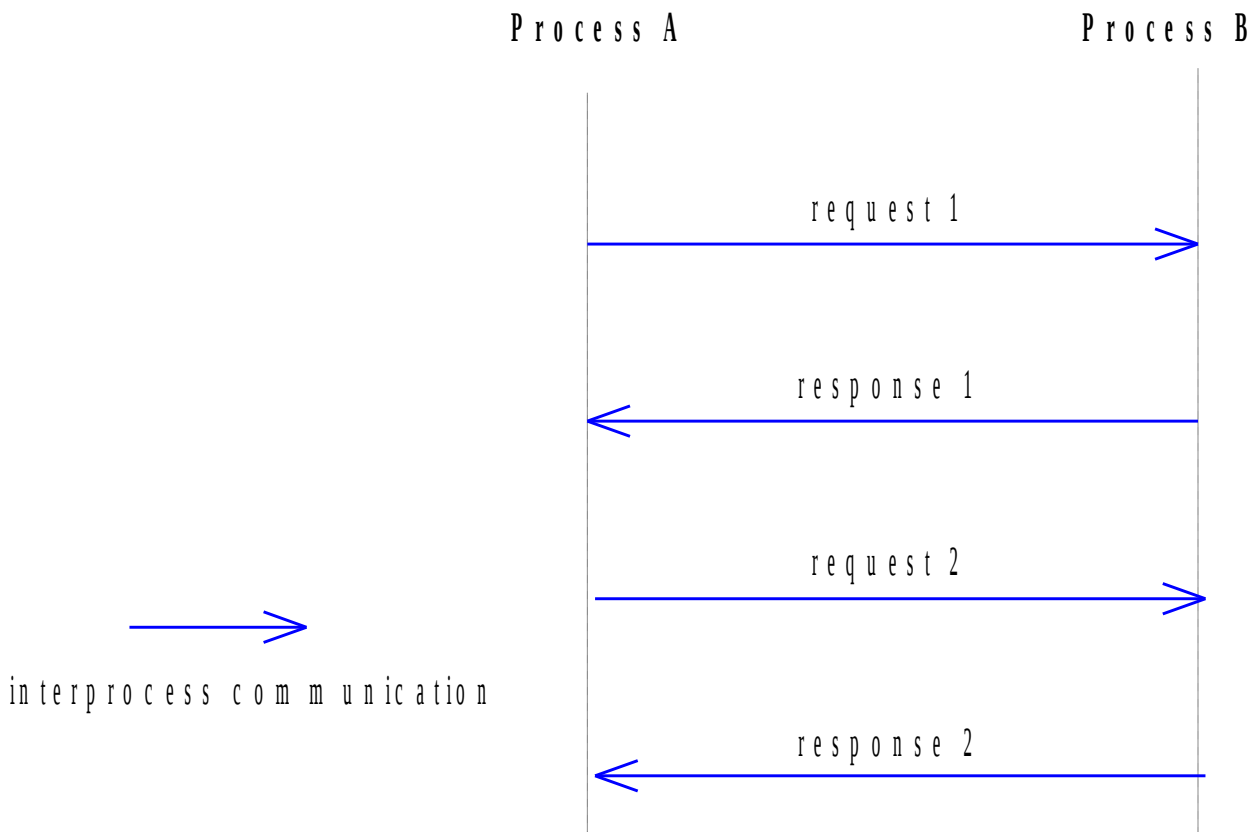
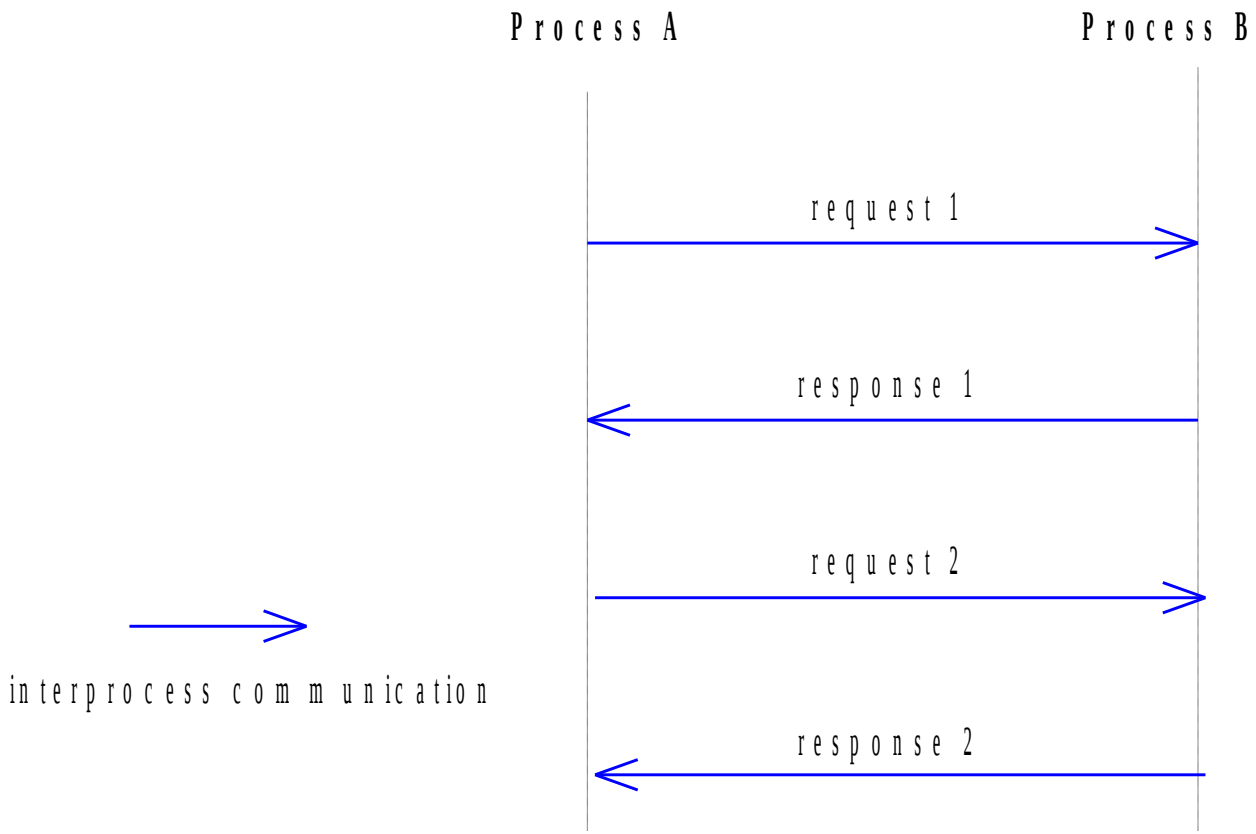


Diagrama de Sequência sessão HTTP



Protocolo

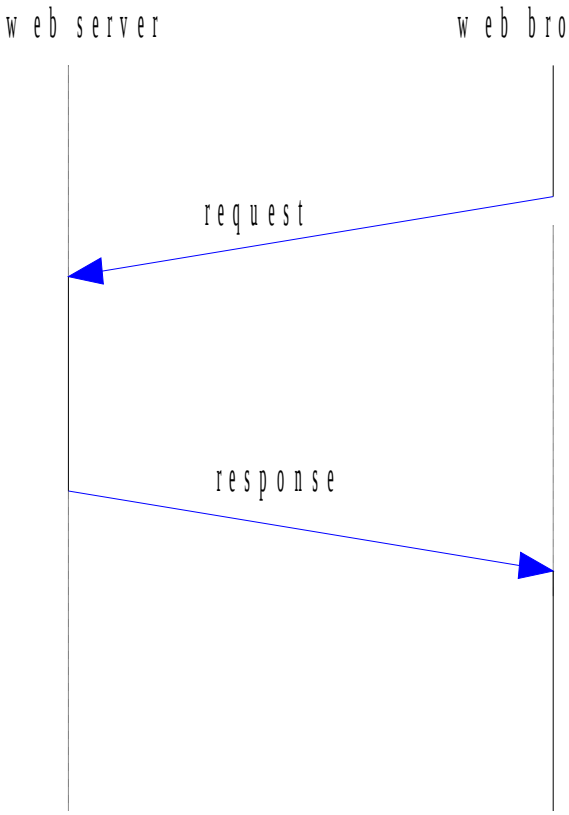
- Em uma aplicação distribuída, dois processos realizam comunicação entre processos com uma concordância mútua em um protocolo.
- A especificação de um protocolo deve incluir (i) a sequência de dados trocados, que pode ser descrita usando um diagrama de eventos no tempo.

  (ii) a especificação do formato dos dados trocados a cada passo.

HTTP: uma amostra de protocolo

- O Hypertext Transfer Protocol é um protocolo para um processo (browser) obter um documento de um processo servidor web.
- É um protocolo requisição/resposta: um browser envia uma requisição para um processo servidor web, que responde com uma resposta.

Protocolo HTTP



request is a message in 3 parts:
- <command> <document adddress> <HTTP version>
- an optional header
- optional data for CGI data using post method

response is a message consisting of 3 parts:
- a status line of the format <protocol><status code><description>
- header information, which may span several lines;
- the document itself.

We will explore HTTP in details later this quarter.

Sessão HTTP

```
Script started on Tue Oct 10 21:49:28 2000
9:49pm telnet www.csc.calpoly.edu 80
Trying 129.65.241.20...
Connected to tiedye2-srv.csc.calpoly.edu.
Escape character is '^]'.
GET   /~mliu/   HTTP/1.0
```

HTTP Request

```
HTTP/1.1 200 OK
Date: Wed, 11 Oct 2000 04:51:18 GMT
Server: Apache/1.3.9 (Unix) ApacheJServ/1.0
Last-Modified: Tue, 10 Oct 2000 16:51:54 GMT
ETag: "1dd1e-e27-39e3492a"
Accept-Ranges: bytes
Content-Length: 3623
Connection: close
Content-Type: text/html
```

HTTP response status line

HTTP response header

```
<HTML>
<HEAD>
<TITLE> Mei-Ling L. Liu's  Home Page
</TITLE>
</HEAD>
<BODY bgcolor=#ffffff>
```

document content

IPC: paradigmas e implementações

Paradigmas de IPC em diferentes níveis de abstração, evolução com correspondentes implementações.