Comunicação entre tarefas

Prof. Paulo Roberto O. Valim
Universidade do Vale do Itajaí

introdução

- **É comum que implementações de sistemas sejam estruturadas na forma de várias tarefas interdependentes que cooperam entre si para atingir os objetivos da aplicação (ex.: navegador web).**

- **Algumas razões para isto:**
  - Atender vários usuários simultâneos;
  - Uso de computadores multiprocessados;
  - Modularidade;
  - Construção de aplicações interativas.

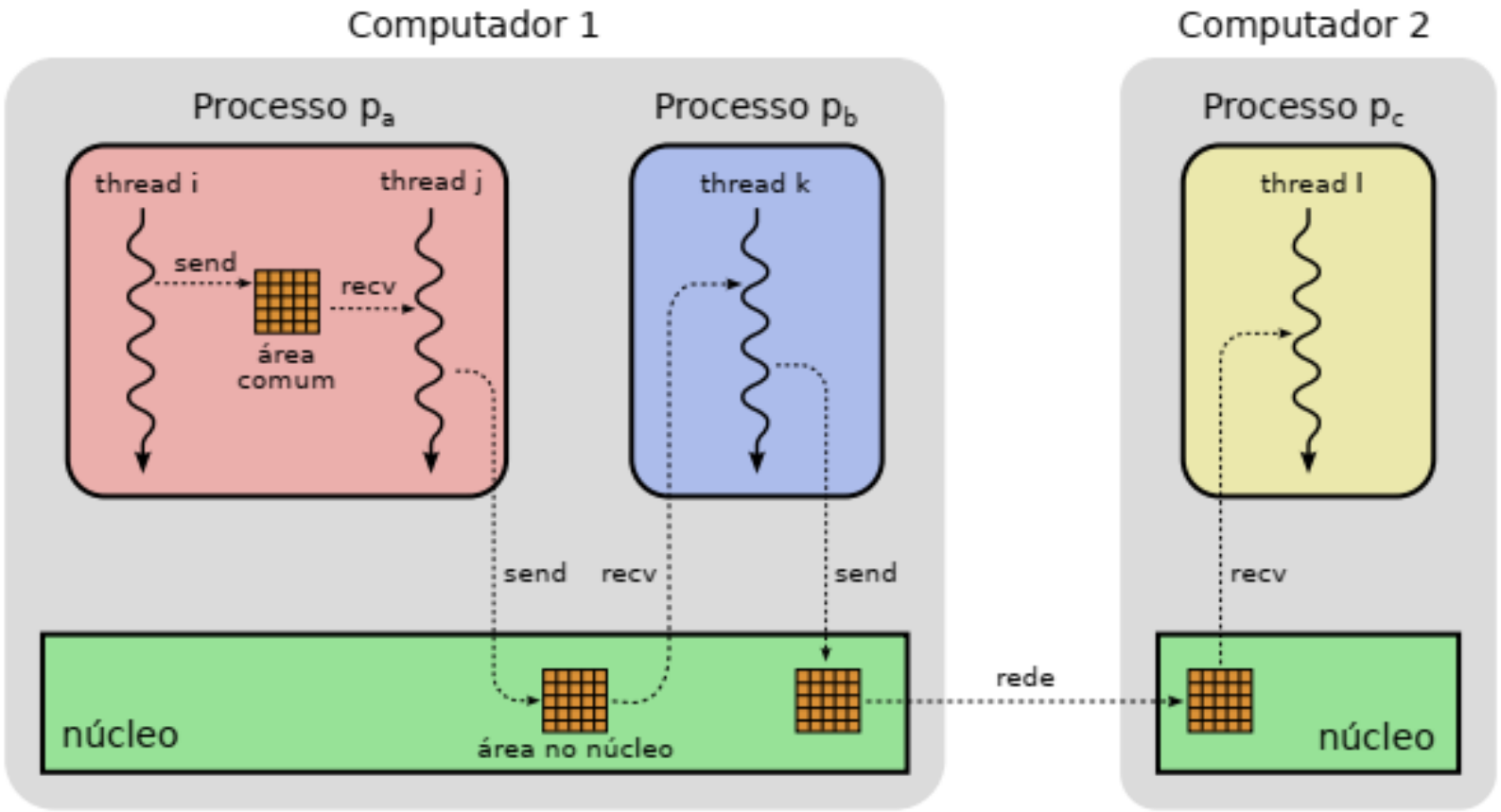
- **Para que a cooperação aconteça, as tarefas precisam se comunicar (trocar informações/dados).**

Escopo da comunicação

- **Os mecanismos de comunicação são denominados de forma genérica como "mecanismos de IPC" *(Inter-process Communication Mechanisms).***

- **Podem ser de três tipos, dependendo da situação:**
  - Comunicação intra-processo;
  - Comunicação inter-processos;
  - Comunicação inter-sistemas;

Escopo da comunicação



Características dos mecanismos de comunicação

- **A implementação da comunicação entre tarefas pode ocorrer de várias formas. Ao definir os mecanismos de comunicação oferecidos por um sistema operacional, seus projetistas devem considerar muitos aspectos, como o formato dos dados a transferir, o sincronismo exigido nas comunicações, a necessidade de buffers e o número de emissores/receptores envolvidos em cada ação de comunicação.**
  - Comunicação direta ou indireta
  - Sincronismo
  - Formato de envio
  - Capacidade dos canais
  - Confiabilidade dos canais
  - Número de participantes

Comunicação direta ou indireta

- **Comunicação direta:** o emissor identifica claramente o receptor e vice-versa. Pode ser implementada por duas primitivas básicas: enviar (dados, destino), que envia os dados relacionados ao destino indicado, e receber (dados, origem), que recebe os dados previamente enviados pela origem indicada.

- **Comunicação indireta:** o emissor e receptor não precisam se conhecer, pois não interagem diretamente entre si. Eles se relacionam através de um canal de comunicação, que é criado pelo sistema operacional, geralmente a pedido de uma das partes. As primitivas de comunicação não designam diretamente tarefas, mas canais de comunicação aos quais as tarefas estão associadas: enviar (dados, canal) e receber (dados, canal).

Comunicação direta ou indireta

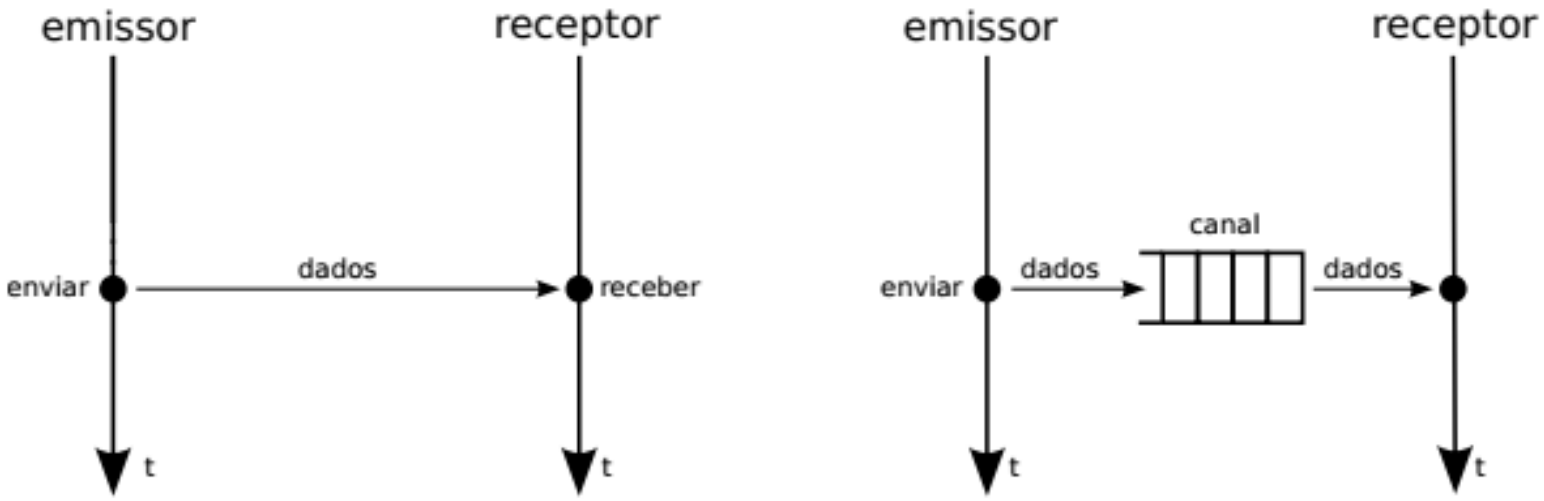


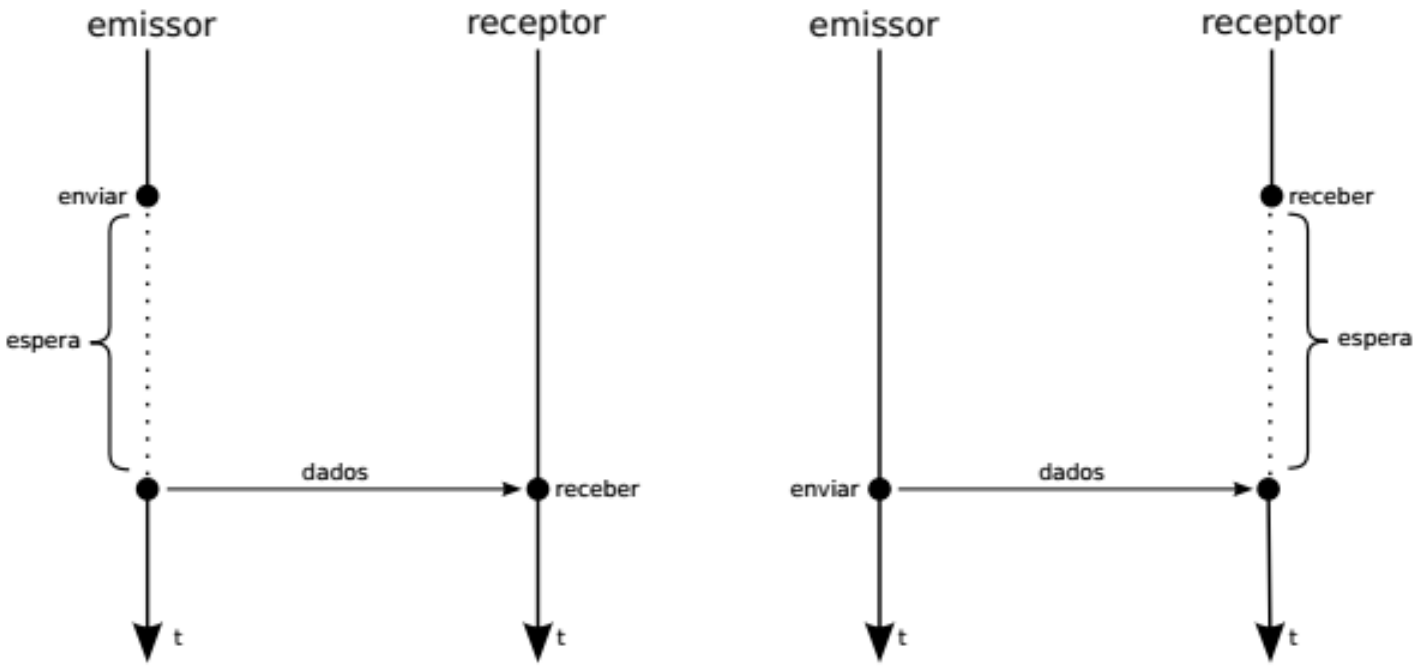
Figura 3.2: Comunicação direta (esquerda) e indireta (direita).

Sincronismo

- **Com relação ao sincronismo, a comunicação pode ser classificada em:**
  - **Síncrona (ou bloqueante):** quando as operações de envio e recepção de dados bloqueiam (suspendem) as tarefas envolvidas até a conclusão da comunicação: o emissor será bloqueado até que a informação seja recebida pelo receptor, e vice-versa.
  - **Assíncrona (ou não bloqueante):** em um sistema com comunicação assíncrona, as primitivas de envio e recepção não são bloqueantes: caso a comunicação não seja possível no momento em que cada operação é invocada, esta retorna imediatamente com uma indicação de erro. Deve-se observar que, caso o emissor e o receptor operem ambos de forma assíncrona, torna-se necessário criar um canal ou buffer para armazenar os dados da comunicação entre eles.
  - **Semissíncrona (ou semibloqueante):** primitivas de comunicação semissíncronas têm um comportamento síncrono (bloqueante) durante um prazo predefinido. Caso esse prazo se esgote sem que a comunicação tenha ocorrido, a primitiva se encerra com uma indicação de erro.

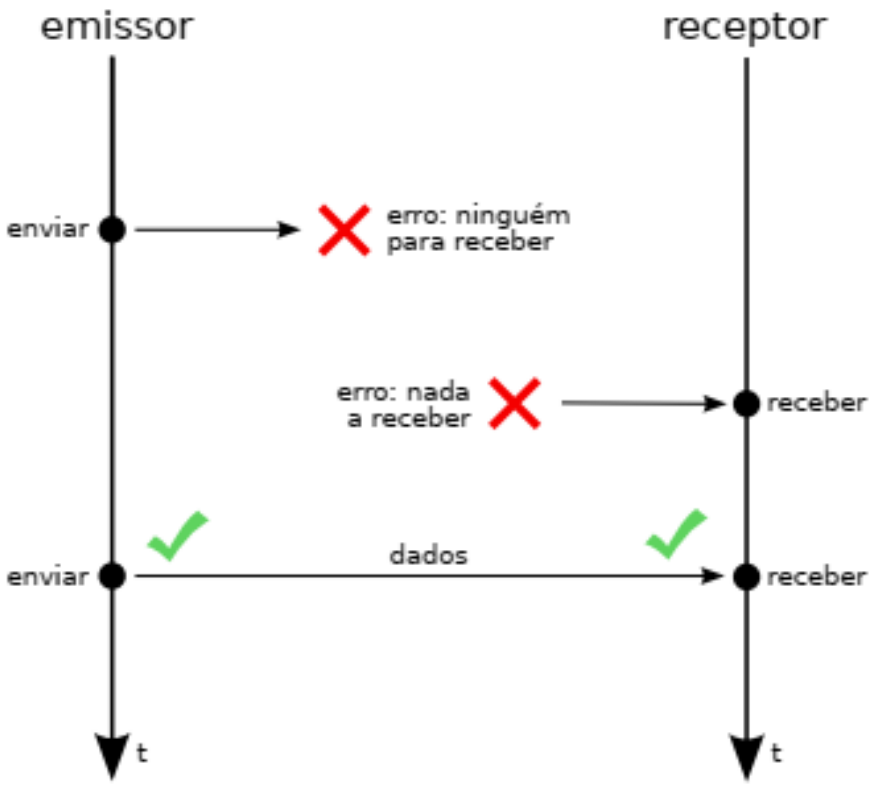
Sincronismo

- **Comunicação síncrona**



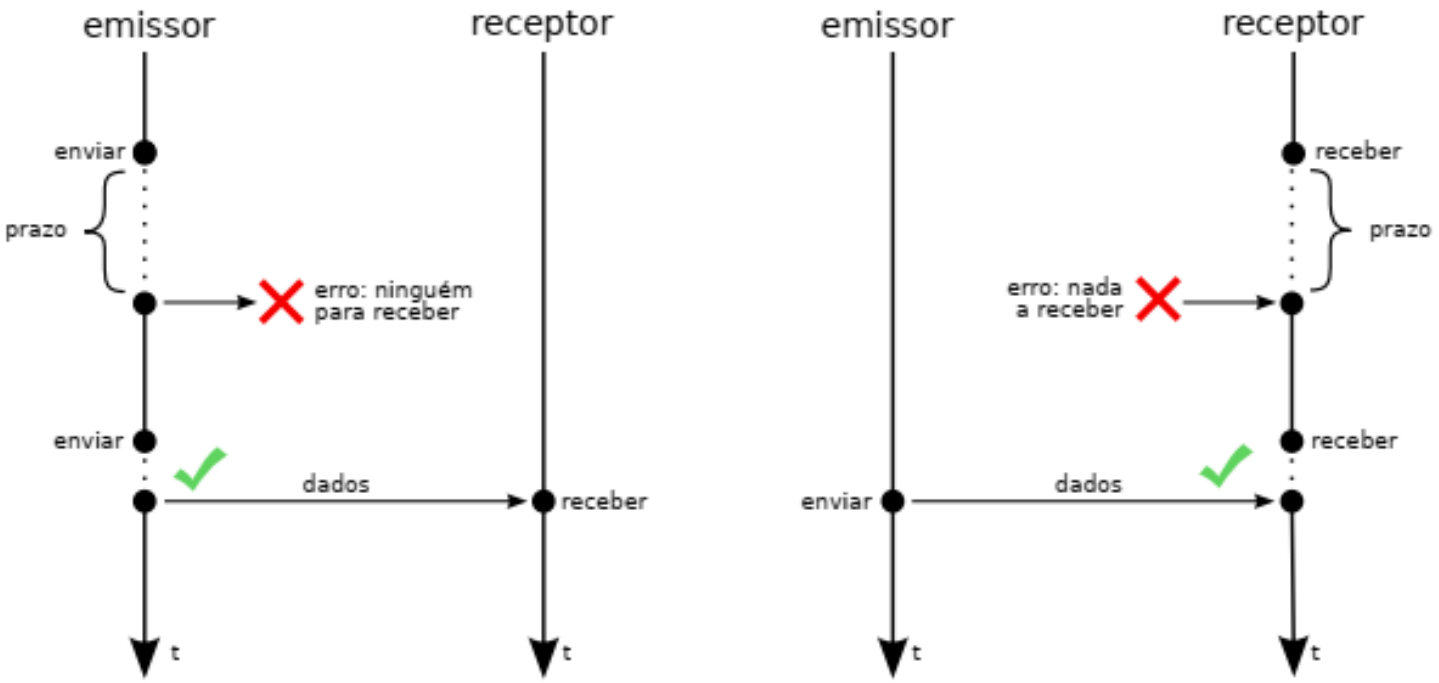
Sincronismo

Comunicação Assíncrona



Sincronismo

Comunicação semissíncrona

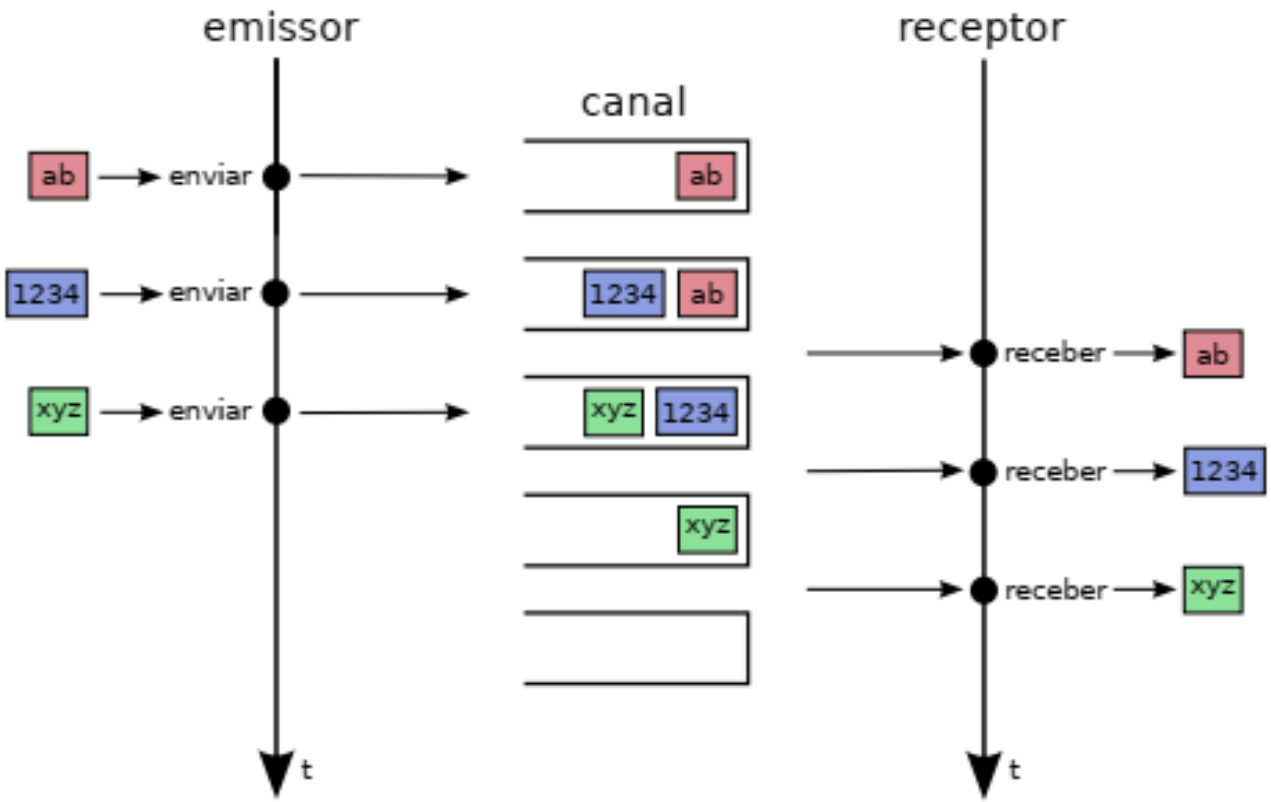


Formato do envio

- **Baseada em mensagens**
  - uma **sequência de mensagens** independentes, cada uma com seu próprio conteúdo
  - cada mensagem consiste de um pacote de dados que pode ser tipado ou não. Esse pacote é recebido ou descartado pelo receptor em sua íntegra; não existe a possibilidade de receber "meia mensagem"
  - Exemplos de sistema de comunicação orientados a mensagens incluem as *message queues* do UNIX e os protocolos de rede IP e UDP

Formato do envio

- **Baseada em mensagens**

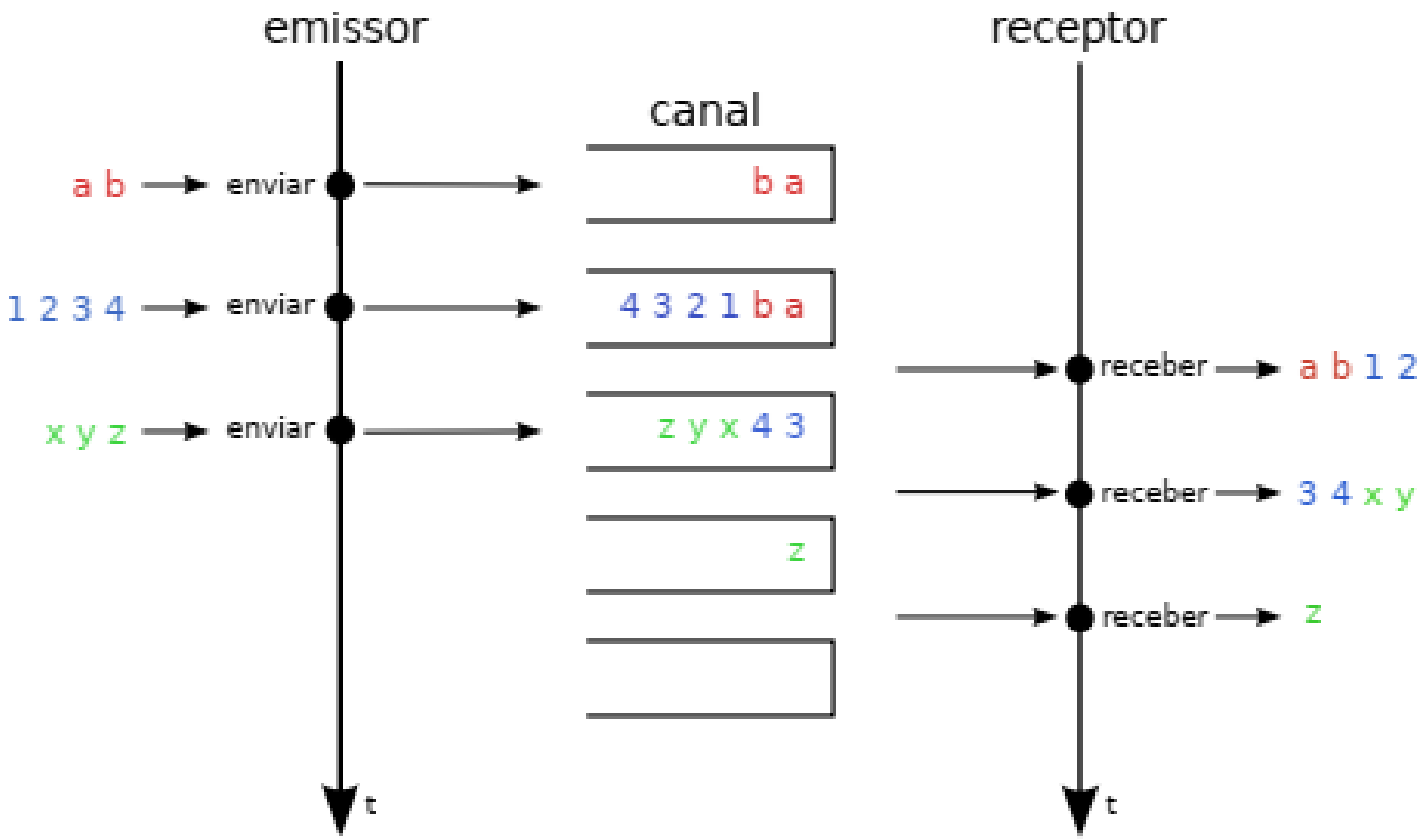


Formato do envio

- **Baseada em fluxo**
  - um **fluxo sequencial** e contínuo de dados, imitando o comportamento de um arquivo com acesso sequencial.
  - Neste caso, o canal de comunicação é visto como o equivalente a um arquivo: o emissor “escreve” dados nesse canal, que serão “lidos” pelo receptor respeitando a ordem de envio dos dados. Não há separação lógica entre os dados enviados em operações separadas: eles podem ser lidos byte a byte ou em grandes blocos a cada operação de recepção, a critério do receptor.
  - Exemplos de sistemas de comunicação orientados a fluxo de dados incluem os *pipes* do UNIX e o protocolo de rede TCP/IP (este último é normalmente classificado como *orientado a conexão*, com o mesmo significado)

Formato do envio

- **Baseada em fluxo**



Capacidade dos canais

Capacidade nula (n=0)

- Neste caso, o canal não pode armazenar dados; a comunicação é feita por transferência direta dos dados do emissor para o receptor, sem cópias intermediárias. Caso a comunicação seja síncrona, o emissor permanece bloqueado até que o destinatário receba os dados, e vice-versa. Essa situação específica (comunicação síncrona com canais de capacidade nula) implica em uma forte sincronização entre as partes, sendo por isso denominada Rendez-Vous (termo francês para “encontro”).

Capacidade dos canais

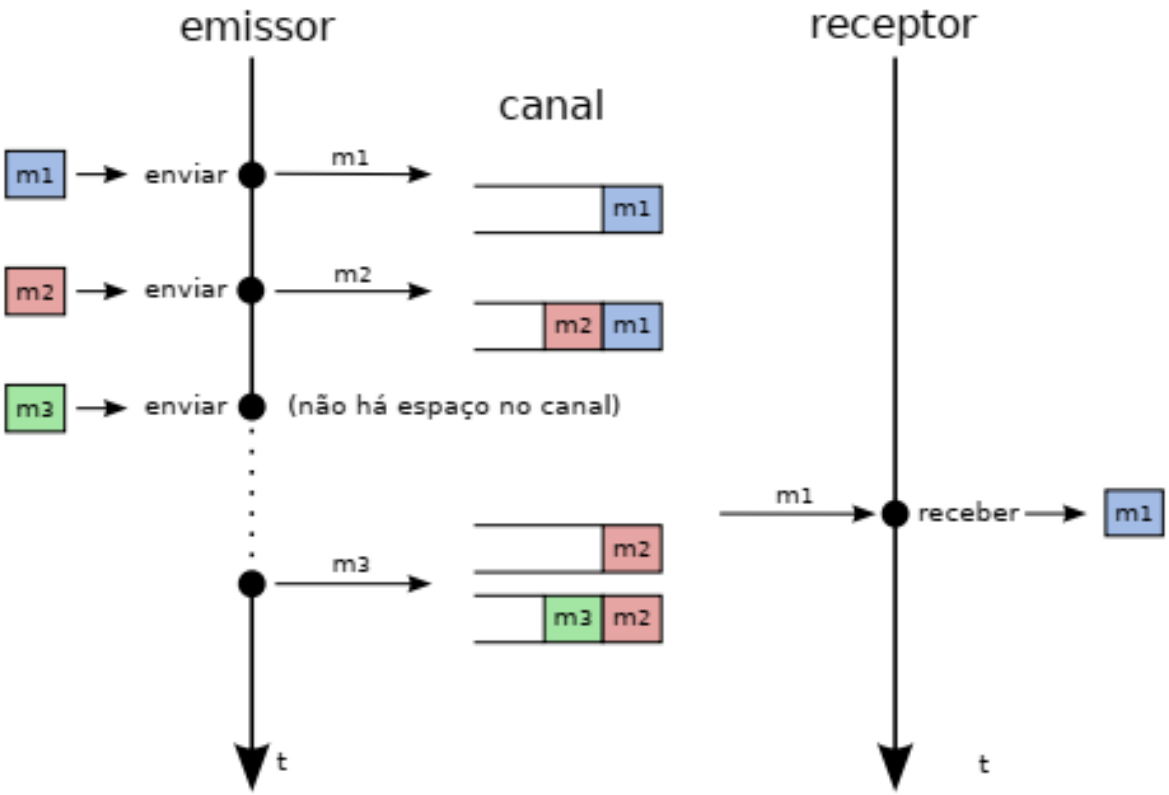
- **Capacidade infinita (n=∞)**
  - O emissor sempre pode enviar dados, que serão armazenados no *buffer* do canal enquanto o receptor não os consumir. Obviamente essa situação não existe na prática, pois todos os sistemas de computação têm capacidade de memória e de armazenamento finitas. No entanto, essa simplificação é útil no estudo dos algoritmos de comunicação e sincronização, pois torna menos complexas a modelagem e análise dos mesmos.

Capacidade dos canais

- **Capacidade finita (0 < n < ∞)**
  - Neste caso, uma quantidade finita (*n*) de dados pode ser enviada pelo emissor sem que o receptor os consuma. Todavia, ao tentar enviar dados em um canal já saturado, o emissor poderá ficar bloqueado até surgir espaço no buffer do canal e conseguir enviar (comportamento síncrono) ou receber um retorno indicando o erro (comportamento assíncrono). A maioria dos sistemas reais opera com canais de capacidade finita

Capacidade dos canais

- **Comunicação bloqueante usando um canal com capacidade 2**



Confiabilidade dos canais

- Quando um canal de comunicação transporta todos os dados enviados através dele para seus receptores, respeitando seus valores e a ordem em que foram enviados, ele é chamado de **canal confiável**. Caso contrário, trata-se de um canal **não-confiável**.

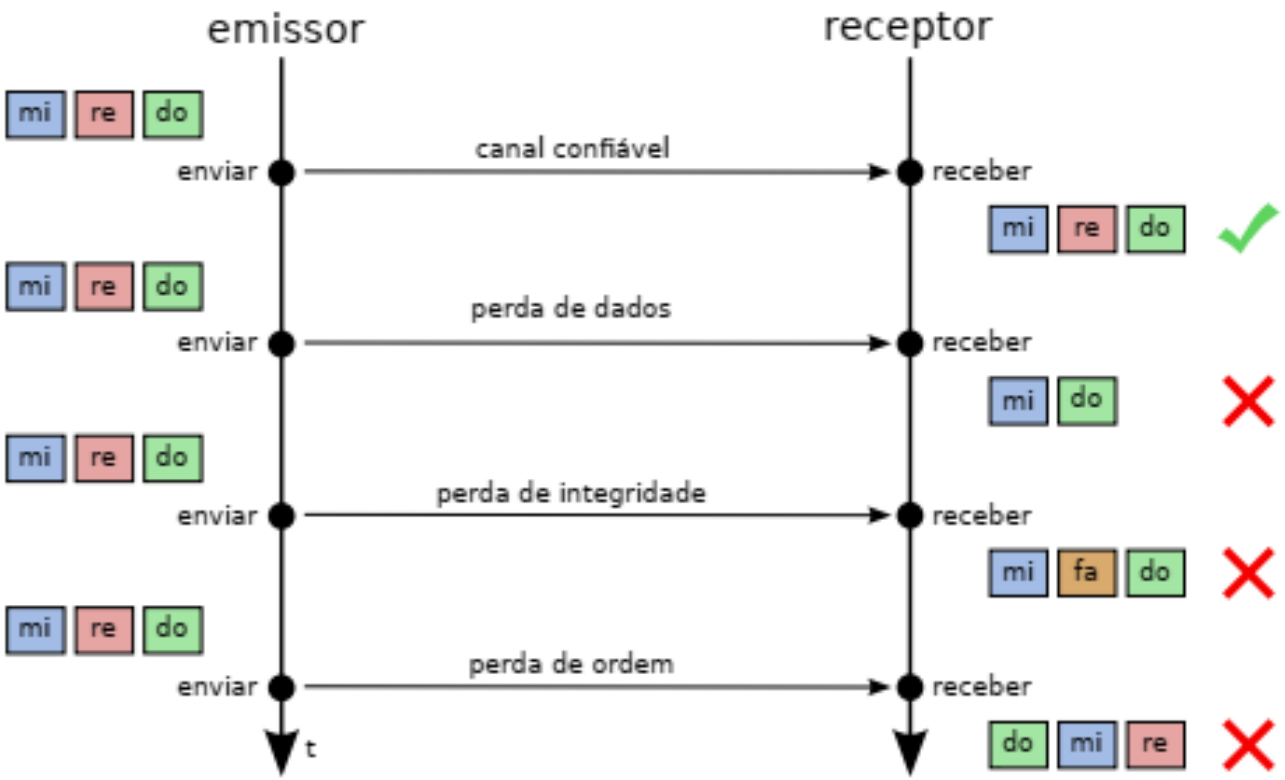
- **Perda de dados**: nem todos os dados enviados através do canal chegam ao seu destino; podem ocorrer perdas de mensagens (no caso de comunicação orientada a mensagens) ou de sequências de bytes, no caso de comunicação orientada a fluxo de dados.

- **Perda de integridade**: os dados enviados pelo canal chegam ao seu destino, mas podem ocorrer modificações em seus valores devido a interferências externas.

- **Perda da ordem**: todos os dados enviados chegam íntegros ao seu destino, mas o canal não garante que eles serão entregues na ordem em que foram enviados. Um canal em que a ordem dos dados é garantida é denominado canal FIFO ou canal ordenado.

Confiabilidade dos canais

- **Comunicação com canais não confiáveis**



Número de participantes

- Nas situações de comunicação apresentadas até agora, cada canal de comunica- ção envolve apenas um emissor e um receptor. No entanto, existem situações em que uma tarefa necessita comunicar com várias outras, como por exemplo em sistemas de *chat* ou mensagens instantâneas (IM – *Instant Messaging*).

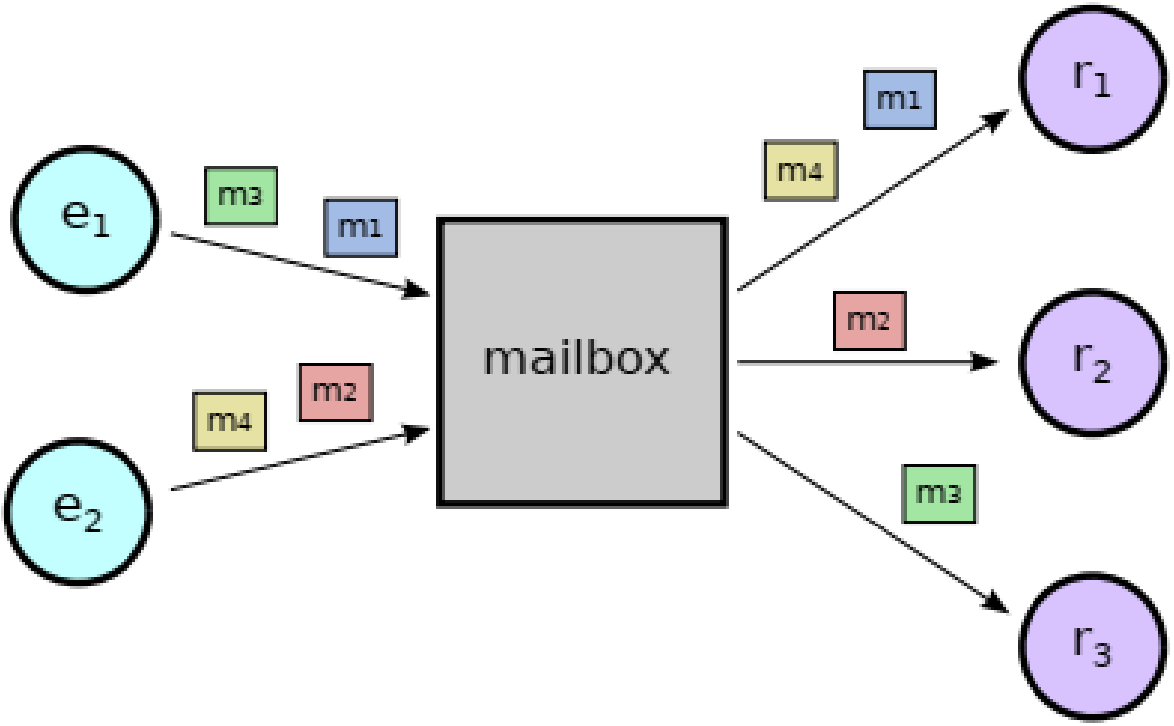
- **1:1** → quando exatamente um emissor e um receptor interagem através do canal de comunicação; é a situação mais frequente, implementada por exemplo nos pipes UNIX e no protocolo TCP.

Número de participantes

- **M:N** → quando um ou mais emissores enviam mensagens para um ou mais receptores. Duas situações distintas podem se apresentar neste caso:
  - Cada mensagem é recebida por apenas um receptor (em geral aquele que pedir primeiro) Essa abordagem é conhecida como ***mailbox***, sendo implementada nas ***message queues*** do UNIX e Windows e também nos ***sockets*** do protocolo UDP.
  - Cada mensagem é recebida por vários receptores (cada receptor recebe uma cópia da mensagem). Essa abordagem é conhecida como ***barramento de mensagens (message bus)***, ***canal de eventos*** ou ainda ***canal publish-subscribe.***

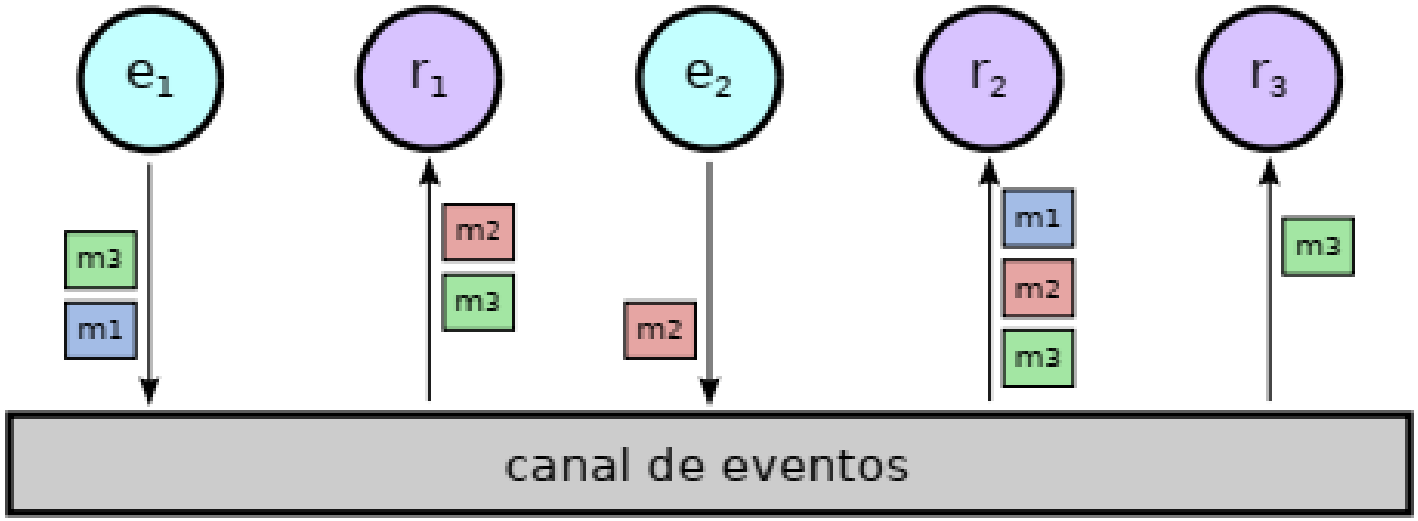
Número de participantes

- Comunicação M:N através de mailbox



Número de participantes

- **Comunicação M:N através de um barramento de mensagens**



Exercício

- Usando o recurso de Pipes do Linux, implementar em “chat” entre processos executando em uma mesma máquina. Para testar execute o programa em dois terminais separados.

Comunicação entre tarefas

- **Referências:**
  - Maziero, C – “Sistemas Operacionais: Conceitos e Mecanismos” – cap 8; (disponível na internet - Versão compilada em 22 de setembro de 2023).
  - Silberschatz, A. – “Operating System Concepts”; 9ª. Edição; editora Wiley – cap 3, tópicos 4, 5 e 6.