UNIVERSIDAD DE COSTA RICA

HPC: PARALELISMO DE MEMORIA DISTRIBUIDA COMUNICACIÓN DE PUNTO A PUNTO

Prof. Marlon Brenes y Prof. Federico Muñoz
Escuela de Física, Universidad de Costa Rica

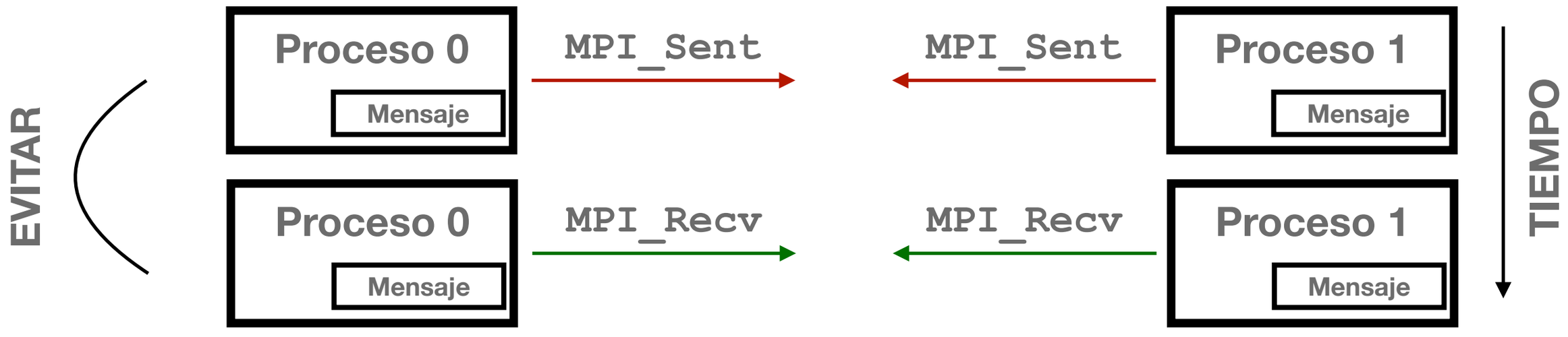
EFis Escuela de Física

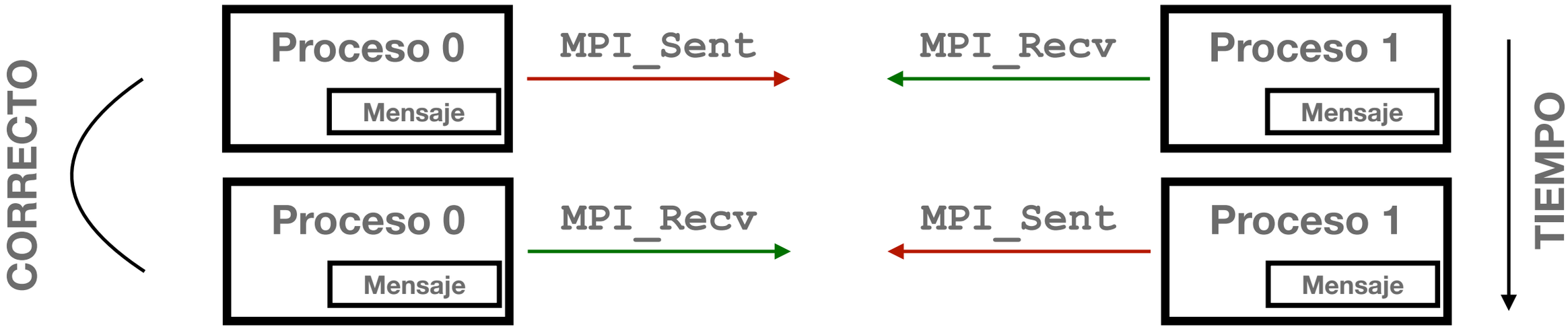
Point-to-point communication: Blocking calls

- Otra operación fundamental de MPI son operaciones de envío y recepción de mensajes <u>con bloqueo</u>
    - Estas funciones no devuelven un valor (no finalizan) hasta que el mensaje no ha sido enviado o recibido

- La regla fundamental es la siguiente:
    - Por cada `MPI_Sent` debe existir un `MPI_Recv` **paralelo en el tiempo**
    - Lo que puede suceder si se viola esta regla depende de la implementación de MPI (pueden ocurrir deadlocks)

Point-to-point communication: Blocking calls





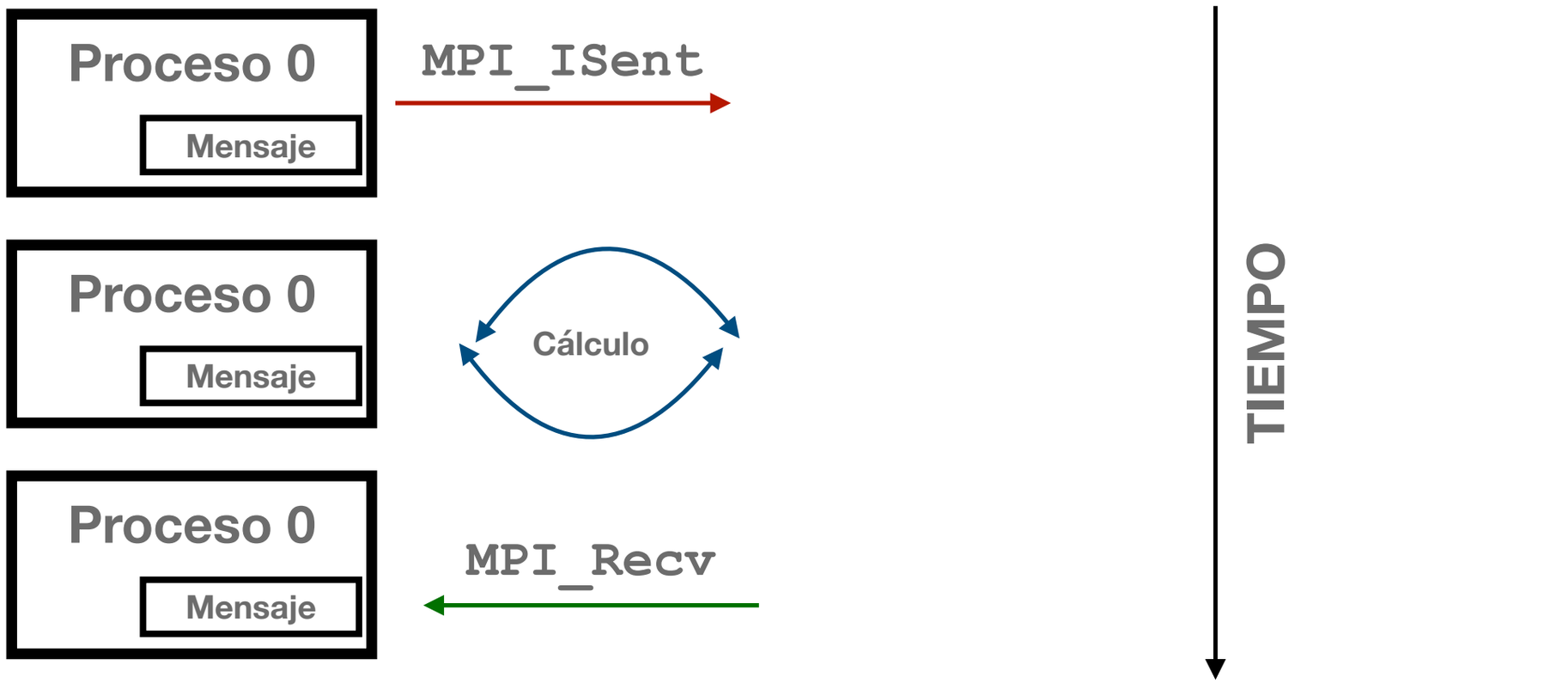
EFis Escuela de Física

Point-to-point communication: Blocking calls

- **Ver:** `order.cpp`

Point-to-point communication: Non-blocking calls

- Existen operaciones que no bloquean el llamado al resultado de la operación, es decir, devuelven un valor de retorno inmediatamente
    - Esto implica que al final del llamado puede no ser seguro accesar la región de memoria de envío/recepción
    - Se utiliza para traslapar paso de mensajes con cálculos
    - **Estas rutinas pueden ser muy poco seguras**

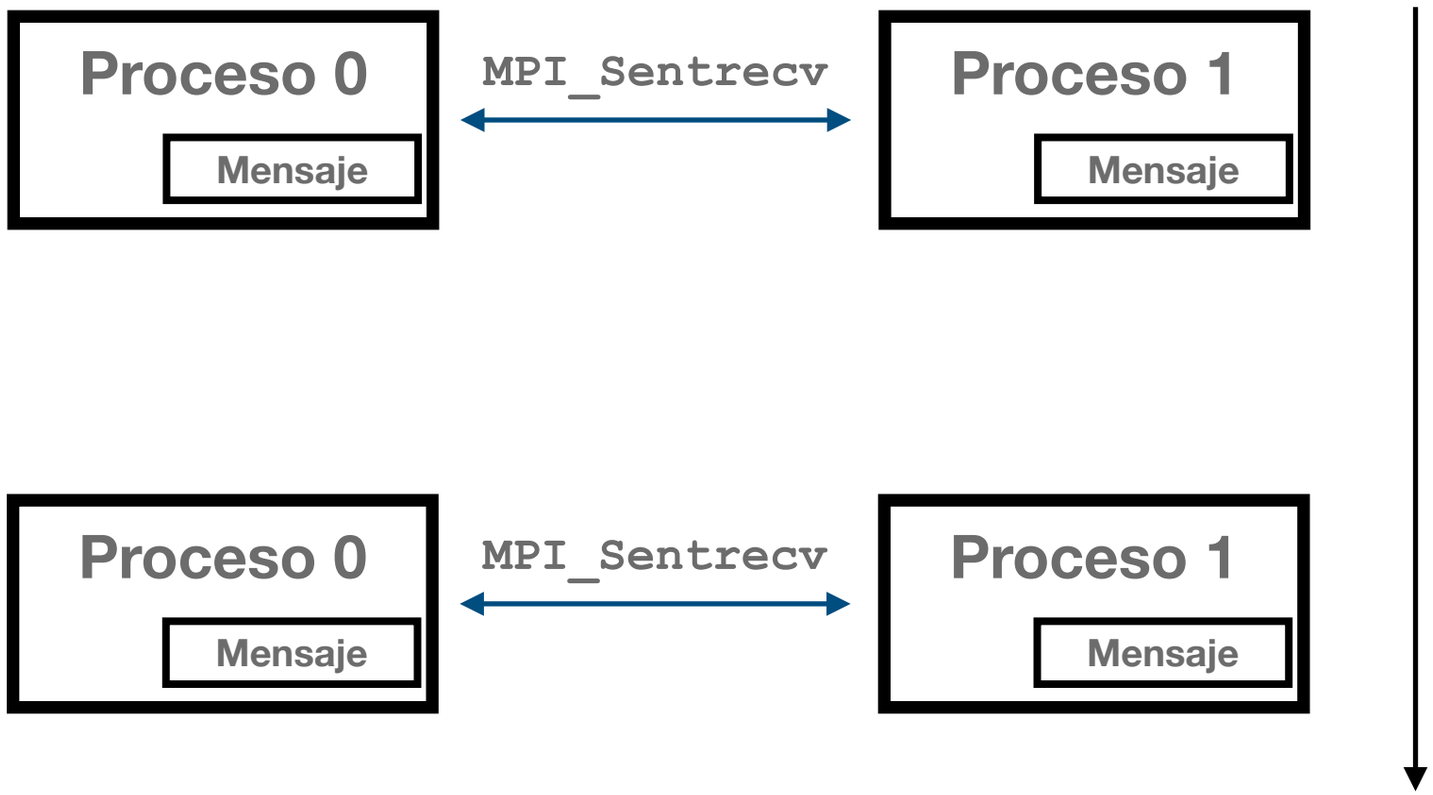


Point-to-point communication: Blocking calls

- Ver: `nonblocking.cpp`

La forma más limpia: `MPI_Sendrecv`

- La mejor forma de realizar comunicaciones de punto a punto es agrupando la operación de envío con la operación de recepción
    - Es menos flexible con respecto al orden de las operaciones
    - Sin embargo, **es la forma más segura**



Laboratorio

- Encontrar en la API sobre: `MPI_Sendrecv`

- Implemente el patrón de comunicación de `order.cpp` utilizando esta función, **específico para dos procesos**

- Existen dos maneras de realizar esta operación: con flujos de control (identificando el rango y ejecutando la función dependiendo del rango) o definir una variable que depende del rango. En este último caso, basta con un llamado a la función! Intente implementar las dos formas

- Encontrar en la API sobre: `MPI_Sendrecv_replace`

- Realice la misma tarea utilizando el mismo buffer para el mensaje

Laboratorio

- **Ver:** `sendrecv_naive.cpp, sendrecv.cpp, sendrecv_replace.cpp`